

z/OS Communications Server



AnyNet Sockets over SNA

Version 1 Release 2

z/OS Communications Server



AnyNet Sockets over SNA

Version 1 Release 2

Note:

Before using this information and the product it supports, be sure to read the general information under "Appendix C. Notices" on page 53.

First Edition (October 2001)

This edition applies to Version 1 Release 2 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

Address comments to:

IBM Corporation
Software Reengineering
Department G71A/ Bldg 503
Research Triangle Park, North Carolina 27709-9990
U.S.A.

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):

1-800-227-5088

Internet e-mail:

usib2hpd@vnet.ibm.com

World Wide Web:

<http://www.ibm.com/servers/eserver/zseries/zos>

IBMLink:

CIBMORCF at RALVM17

IBM Mail Exchange:

tkinlaw@us.ibm.com

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1997, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	ix
About This Book	xi
Artwork Used in This Book	xi
Where to Find More Information.	xii
Where to Find Related Information on the Internet.	xiii
Licensed Documents.	xiii
LookAt, an Online Message Help Facility	xiv
How to Contact IBM® Service	xv
z/OS Communications Server Information	xv
Summary of Changes	xxiii
Chapter 1. Introducing Sockets over SNA	1
What Does Sockets over SNA Do?	1
How Sockets over SNA Works on z/OS	1
Generating LU 6.2 Calls from Socket Calls	2
Mapping an IP Address to a Fully Qualified LU Name	2
Example Network Scenarios that Include Sockets over SNA.	3
Communicating from VTAM Subarea to VTAM Subarea	3
Communicating from VTAM Subarea to OS/2 APPC through a 3172 Interconnect Controller.	3
Communicating from VTAM Subarea to OS/2 APPC through NCP/NTRI	4
System Software Required to Use AnyNet Sockets over SNA	5
Applications Supported by AnyNet Sockets over SNA	5
z/OS UNIX System Services Socket Applications Supported by Sockets over SNA	5
Requirements for Full-duplex Enablement	6
Managing Sockets over SNA	6
Chapter 2. Planning for AnyNet Sockets over SNA	7
Assigning IP Addresses to Sockets-over-SNA Nodes	7
Assigning SNA Addresses to Sockets-over-SNA Nodes	8
Assigning SNA Network Names to Sockets-over-SNA Nodes	8
Assigning SNA LU Names to Sockets-over-SNA Nodes	8
Mapping IP Addresses to SNA Addresses	9
Generating an SNA Address from an IP Address	10
Example of an Entry in the IP-LU Mapping Table	11
How Sockets over SNA Uses IP-LU Mapping Table Entries.	11
Summary of the IP-LU Mapping Process	12
Assigning an IP Address to the Sockets-over-SNA Interface	12
Domain Name Server	13
Defining ETC Hosts, Networks, Protocols, and Services to z/OS UNIX System Services	13
Chapter 3. Defining AnyNet Sockets over SNA	15
Defining AnyNet Sockets over SNA to VTAM	15
Defining APPL Statements for Sockets over SNA	15
Increasing the Session Limit for Sockets over SNA	15
Defining a MODEENT Macro Entry for Sockets over SNA	16
Defining Remote Nodes to Sockets over SNA	17

Using the ISTSKMAP Utility to Establish, Update, and Query the Mapping Table	18
Syntax of the ISTSKMAP Utility	18
Using the Convert Option to Generate LU Names	21
Using the ISTSKIFC Utility to Define an SNA Network Interface for Sockets over SNA	22
Using the ISTSKRTE Utility to Update the Local Sockets over SNA Routing Table.	23
 Chapter 4. Configuring, Customizing, and Tuning AnyNet Sockets over SNA	27
Defining Sockets-over-SNA Environment Variables: the ENVVAR Data Set	27
CONNECT_TIMEOUT	28
DG_CONV_IDLE_TIMEOUT	28
DG_SEGMENTATION	28
HOSTNAME	28
LEARNMAP = NOIYES	29
MAX_CONCURRENT_SOCKET_CALLS	29
MAX_DG_CONVS	30
MAX_SENDBUF	30
OE_INADDRANY_COUNT	30
OE_INADDRANY_PORT	31
SXMODE_DEFAULT	31
SXMODEn	31
TRACE_OPTION	32
Routing ENVVAR Messages	32
Customizing Message Text for Sockets-over-SNA Messages	32
Defining the Sockets-over-SNA Transport Driver to z/OS UNIX System Services Restrictions and Constraints	32
 Chapter 5. Starting, Initializing, and Stopping AnyNet Sockets over SNA	35
Order of Initialization	35
Starting and Initializing Sockets over SNA	35
Running Utilities Using JCL	37
Running Utilities from the z/OS UNIX Shell	37
Stopping Sockets over SNA	38
 Chapter 6. Diagnosing Problems for AnyNet Sockets over SNA	39
Determining Errors During Sockets-over-SNA Initialization and Operation	39
Determining Whether Environment Variables Are Set Successfully	40
Using the ISTSKNST Utility to Display Sockets-over-SNA Routing Tables	40
Using the OPING Utility.	42
Using the ISTSKTRC Utility to Trace Sockets over SNA	43
Syntax for Running ISTSKTRC	43
Routing Output from the ISTSKTRC Utility	44
Responding to Sockets-over-SNA Abends	45
Service Information	45
 Appendix A. Syntax Conventions	47
How to Read a Syntax Diagram	47
Symbols and Punctuation	47
Parameters	47
Syntax Examples	48
 Appendix B. Information Apars	51
IP Information Apars	51
SNA Information Apars	52

Appendix C. Notices 53

Programming Interfaces 55

Trademarks 56

Index 59

Figures

1.	Conventions Used in Network Illustrations.	xii
2.	Structure of an AnyNet Sockets over SNA Node.	2
3.	z/OS UNIX Socket Applications Communicating from VTAM Subarea to VTAM Subarea	3
4.	Socket Applications Communicating from VTAM Subarea to OS/2 APPC through a 3172 Interconnect Controller	4
5.	Socket Applications Communicating from VTAM Subarea to OS/2 APPC through NCP/NTRI	5

Tables

1.	Example of an Entry in the IP-LU Mapping Table	11
2.	Limitations for Defining LU Templates	21
3.	Determining the Classification of an IP Address	23
4.	IP Information Apars	51
5.	SNA Information Apars.	52

About This Book

This publication describes the primary tasks involved with installing, configuring, using, and diagnosing AnyNet[®] Sockets over SNA.

This guide is intended for system programmers who require information to help install, configure use, and diagnose Sockets over SNA.

Artwork Used in This Book

Figure 1 on page xii shows the conventions used in this book to illustrate the parts of a network.

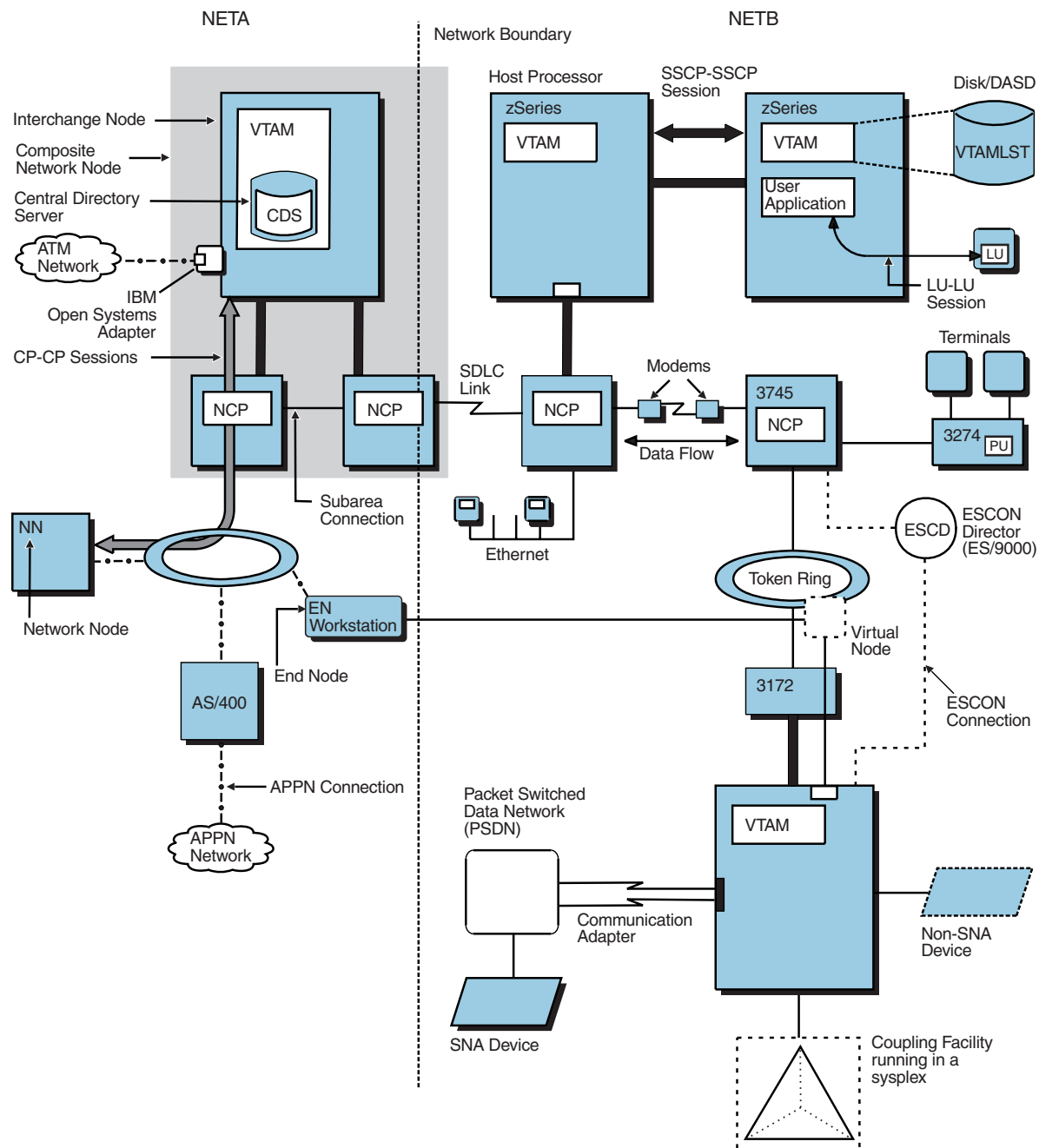


Figure 1. Conventions Used in Network Illustrations

Where to Find More Information

This section contains:

- Pointers to information available on the Internet
- Information about licensed documentation
- Information about LookAt, the online message tool

- A set of tables that describes the books in the z/OS Communications Server (z/OS CS) library, along with related publications

Where to Find Related Information on the Internet

Home Page	Web address
z/OS	http://www.ibm.com/servers/eserver/zseries/zos/
z/OS Internet Library	http://www.ibm.com/servers/eserver/zseries/zos/bkserv/
IBM Communications Server product	http://www.software.ibm.com/network/commserver/
IBM Communications Server support	http://www.software.ibm.com/network/commserver/support/
IBM Systems Center publications	http://www.redbooks.ibm.com/
IBM Systems Center flashes	http://www-1.ibm.com/support/techdocs/atsmastr.nsf
VTAM and TCP/IP	http://www.software.ibm.com/network/commserver/about/csos390.html
IBM	http://www.ibm.com
RFC	http://www.ietf.org/rfc.html

Information about Web addresses can also be found in informational APAR II11334.

DNS Web Sites

For information about DNS, see the following Web sites:

USENET news groups:

comp.protocols.dns.bind

For BIND mailing lists, see:

- <http://www.isc.org/ml-archives/>
 - BIND Users
 - Subscribe by sending mail to bind-users-request@isc.org
 - Submit questions or answers to this forum by sending mail to bind-users@isc.org
 - BIND 9 Users (Note: This list may not be maintained indefinitely.)
 - Subscribe by sending mail to bind9-users-request@isc.org
 - Submit questions or answers to this forum by sending mail to bind9-users@isc.org

For definitions of the terms and abbreviations used in this book, you can view or download the latest *IBM Glossary of Computing Terms* at the following Web address:

<http://www.ibm.com/ibm/terminology>

Note: Any pointers in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Licensed Documents

z/OS Communications Server licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at <http://www.ibm.com/servers/resourceLink>. Licensed books are available only to customers with a z/OS Communications Server license. Access to these books requires an IBM Resource Link Web user ID and password, and a key code. With

your z/OS Communications Server order, you received a memo that includes this key code. To obtain your IBM Resource Link Web user ID and password, log on to <http://www.ibm.com/servers/resourcelink>. To register for access to the z/OS licensed books perform the following steps:

1. Log on to Resource Link using your Resource Link user ID and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code, you will receive confirmation that your request is being processed. After your request is processed, you will receive an e-mail confirmation.

You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed. To access the licensed books:

1. Log on to Resource Link using your Resource Link user ID and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS Communications Server**.
6. Access the licensed book by selecting the appropriate element.

LookAt, an Online Message Help Facility

LookAt is an online facility that allows you to look up explanations for z/OS CS messages and system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on **News and Help** or from the z/OS V1R2 Collection, SK3T-4269.

To find a message explanation from a TSO command line, simply enter **lookat+message ID**, as in the following example:

```
lookat ezz8477i
```

This results in direct access to the message explanation for message EZZ8477I.

You can use LookAt on the Internet at the following Web site:
www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html

To find a message explanation from the LookAt Web site, simply enter the message ID. You can select the release, if applicable.

How to Contact IBM® Service

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

z/OS Communications Server Information

This section contains descriptions of the books in the z/OS Communications Server library.

z/OS Communications Server publications are available:

- Online at the z/OS Internet Library web page at <http://www.ibm.com/servers/eserver/zseries/zos/>
- In hardcopy and softcopy
- In softcopy only

Softcopy Information

Softcopy publications are available in the following collections:

Titles	Order Number	Description
<i>z/OS V1R2 Collection</i>	SK3T-4269	This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R2, in both BookManager and PDF formats.
<i>z/OS Software Products Collection</i>	SK3T-4270	This CD includes, in both BookManager and PDF formats, the libraries of z/OS software products that run on z/OS but are not elements and features, as well as the <i>Getting Started with Parallel Sysplex</i> bookshelf.
<i>z/OS V1R2 and Software Products DVD Collection</i>	SK3T-4271	This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
<i>z/OS Licensed Product Library</i>	SK3T-4307	This CD includes the licensed books in both BookManager and PDF format.
<i>System Center Publication IBM S/390 Redbooks Collection</i>	SK2T-2177	This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

z/OS Communications Server Library

The following abbreviations follow each order number in the tables below.

HC/SC — Both hardcopy and softcopy are available.

SC — Only softcopy is available. These books are available on the CD Rom accompanying z/OS (SK3T-4269 or SK3T-4307). Unlicensed books can be viewed at the z/OS Internet library site.

Updates to books are available on RETAIN and in the document called *OS/390 DOC APARs and ++HOLD DOC data* which can be found at

http://www.s390.ibm.com/os390/bkserv/new_tech_info.html. See “Appendix B. Information Apars” on page 51 for a list of the books and the informational apars (info apars) associated with them.

Planning and Migration:

Title	Number	Format	Description
<i>z/OS Communications Server: SNA Migration</i>	GC31-8774	HC/SC	This book is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.
<i>z/OS Communications Server: IP Migration</i>	GC31-8773	HC/SC	This book is intended to help you plan for TCP/IP Services, whether you are migrating from a previous version or installing IP for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with TCP/IP Services.

Resource Definition, Configuration, and Tuning:

Title	Number	Format	Description
<i>z/OS Communications Server: IP Configuration Guide</i>	SC31-8775	HC/SC	This book describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this book in conjunction with the <i>z/OS Communications Server: IP Configuration Reference</i> .
<i>z/OS Communications Server: IP Configuration Reference</i>	SC31-8776	HC/SC	This book presents information for people who want to administer and maintain IP. Use this book in conjunction with the <i>z/OS Communications Server: IP Configuration Guide</i> . The information in this book includes: <ul style="list-style-type: none"> • TCP/IP configuration data sets • Configuration statements • Translation tables • SMF records • Protocol number and port assignments
<i>z/OS Communications Server: SNA Network Implementation Guide</i>	SC31-8777	HC/SC	This book presents the major concepts involved in implementing an SNA network. Use this book in conjunction with the <i>z/OS Communications Server: SNA Resource Definition Reference</i> .
<i>z/OS Communications Server: SNA Resource Definition Reference</i>	SC31-8778	HC/SC	This book describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this book in conjunction with the <i>z/OS Communications Server: SNA Network Implementation Guide</i> .
<i>z/OS Communications Server: SNA Resource Definition Samples</i>	SC31-8836	SC	This book contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
<i>z/OS Communications Server: AnyNet SNA over TCP/IP</i>	SC31-8832	SC	This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.

Title	Number	Format	Description
<i>z/OS Communications Server AnyNet Sockets over SNA</i>	SC31-8831	SC	This guide provides information to help you install, configure, use, and diagnose sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.

Operation:

Title	Number	Format	Description
<i>z/OS Communications Server: IP User's Guide and Commands</i>	SC31-8780	HC/SC	This book describes how to use TCP/IP applications. It contains requests that allow a user to: log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
<i>z/OS Communications Server: IP System Administrator's Commands</i>	SC31-8781	HC/SC	This book describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.
<i>z/OS Communications Server: SNA Operation</i>	SC31-8779	HC/SC	This book serves as a reference for programmers and operators requiring detailed information about specific operator commands.
<i>z/OS Communications Server: Operations Quick Reference</i>	SX75-0124	HC/SC	This book contains essential information about SNA and IP commands.

Customization:

Title	Number	Format	Description
<i>z/OS Communications Server: SNA Customization</i>	LY43-0092	SC	This book enables you to customize SNA, and includes the following: <ul style="list-style-type: none"> • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines
<i>z/OS Communications Server: IP Network Print Facility</i>	SC31-8833	SC	This book is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Writing Application Programs:

Title	Number	Format	Description
<i>z/OS Communications Server: IP Application Programming Interface Guide</i>	SC31-8788	SC	This book describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this book to adapt your existing applications to communicate with each other using sockets over TCP/IP.
<i>z/OS Communications Server: IP CICS Sockets Guide</i>	SC31-8807	SC	This book is for people who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.
<i>z/OS Communications Server: IP IMS Sockets Guide</i>	SC31-8830	SC	This book is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM's TCP/IP Services.
<i>z/OS Communications Server: IP Programmer's Reference</i>	SC31-8787	SC	This book describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
<i>z/OS Communications Server: SNA Programming</i>	SC31-8829	SC	This book describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
<i>z/OS Communications Server: SNA Programmers LU 6.2 Guide</i>	SC31-8811	SC	This book describes how to use the SNA LU 6.2 application programming interface for host application programs. This book applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this book.)
<i>z/OS Communications Server: SNA Programmers LU 6.2 Reference</i>	SC31-8810	SC	This book provides reference material for the SNA LU 6.2 programming interface for host application programs.
<i>z/OS Communications Server: CSM Guide</i>	SC31-8808	SC	This book describes how applications use the communications storage manager.
<i>z/OS Communications Server: CMIP Services and Topology Agent Guide</i>	SC31-8828	SC	This book describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The book provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis:

Title	Number	Format	Description
<i>z/OS Communications Server: IP Diagnosis</i>	GC31-8782	HC/SC	This book explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
<i>z/OS Communications Server: SNA Diagnosis V1 Techniques and Procedures</i> and <i>z/OS Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT</i>	LY43-0088 LY43-0089	HC/SC	These books help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
<i>z/OS Communications Server: SNA Data Areas Volume 1</i> and <i>z/OS Communications Server: SNA Data Areas Volume 2</i>	LY43-0090 LY43-0091	SC	These books describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and Codes:

Title	Number	Format	Description
<i>z/OS Communications Server: SNA Messages</i>	SC31-8790	HC/SC	This book describes the ELM, IKT, IST, ISU, IUT, IVT, and USS messages. Other information in this book includes: <ul style="list-style-type: none"> • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
<i>z/OS Communications Server: IP Messages Volume 1 (EZA)</i>	SC31-8783	HC/SC	This volume contains TCP/IP messages beginning with EZA.
<i>z/OS Communications Server: IP Messages Volume 2 (EZB)</i>	SC31-8784	HC/SC	This volume contains TCP/IP messages beginning with EZB.
<i>z/OS Communications Server: IP Messages Volume 3 (EZY)</i>	SC31-8785	HC/SC	This volume contains TCP/IP messages beginning with EZY.
<i>z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)</i>	SC31-8786	HC/SC	This volume contains TCP/IP messages beginning with EZZ and SNM.
<i>z/OS Communications Server: IP and SNA Codes</i>	SC31-8791	HC/SC	This book describes codes and other information that appear in z/OS Communications Server messages.

APPC Application Suite:

Title	Number	Format	Description
<i>z/OS Communications Server: APPC Application Suite User's Guide</i>	GC31-8809	SC	This book documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

Title	Number	Format	Description
<i>z/OS Communications Server APPC Application Suite Administration</i>	SC31-8835	SC	This book contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.
<i>z/OS Communications Server: APPC Application Suite Programming</i>	SC31-8834	SC	This book provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Redbooks

The following Redbooks may help you as you implement z/OS Communications Server.

Title	Number
<i>TCP/IP Tutorial and Technical Overview</i>	GG24-3376
<i>SNA and TCP/IP Integration</i>	SG24-5291
<i>IBM Communication Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 1: Configuration and Routing</i>	SG24-5227
<i>IBM Communication Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 2: UNIX Applications</i>	SG24-5228
<i>IBM Communication Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 3: MVS Applications</i>	SG24-5229
<i>OS/390 Secureway Communication Server V2R8 TCP/IP Guide to Enhancements</i>	SG24-5631
<i>TCP/IP in a Sysplex</i>	SG24-5235
<i>Managing OS/390 TCP/IP with SNMP</i>	SG24-5866
<i>Security in OS/390-based TCP/IP Networks</i>	SG24-5383
<i>IP Network Design Guide</i>	SG24-2580

Related Information

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

The table below lists books that may be helpful to readers.

Title	Number
<i>z/OS SecureWay Security Server Firewall Technologies</i>	SC24-5922
<i>S/390: OSA-Express Customer's Guide and Reference</i>	SA22-7403
<i>z/OS MVS Diagnosis: Procedures</i>	GA22-7587
<i>z/OS MVS Diagnosis: Reference</i>	GA22-7588
<i>z/OS MVS Diagnosis: Tools and Service Aids</i>	GA22-7589

Determining If a Publication Is Current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. Here is how to determine if you are looking at the most current copy of a publication:

1. At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
2. If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
3. To compare softcopy publications, you can check the last two characters of the publication's filename (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

Summary of Changes

Summary of Changes for SC31-8831-00 z/OS Version 1 Release 2

This book contains information previously presented in *OS/390 V2R5 eNetwork Communications Server: AnyNet Sockets over SNA*, SC31-8577.

New Information

- SYS1.SAMPLIB member ISTSKDMM now contains sample JCL for compiling and linking with ISTSKMSG. See “Customizing Message Text for Sockets-over-SNA Messages” on page 32.
- New information has been added about routing output from the ISTSKTRC utility. See “Routing Output from the ISTSKTRC Utility” on page 44.

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Introducing Sockets over SNA

This chapter describes:

- How Sockets over SNA works on z/OS
- Sample Sockets-over-SNA network scenarios, illustrating how Sockets-over-SNA nodes can communicate in SNA networks
- System software required to use Sockets over SNA
- System hardware required to use Sockets over SNA
- Restrictions on using Sockets over SNA
- Application program support provided by Sockets over SNA
- How Sockets over SNA and TCP/IP coexist on the same node
- How the IBM NetView[®] family of products can be used to manage a Sockets-over-SNA network.

What Does Sockets over SNA Do?

Sockets over SNA is one of IBM's AnyNet implementations. AnyNet software enables application programs to communicate over different transport networks and across interconnected networks. Using AnyNet, you can reduce the number of transport networks and reduce operational complexity. These benefits are gained without modification to your existing application programs or hardware.

How Sockets over SNA Works on z/OS

Sockets over SNA provides SNA transport to z/OS UNIX System Services (z/OS UNIX) socket application programs. When a z/OS UNIX application program issues a socket API call, z/OS UNIX provides Sockets over SNA with the information necessary to generate the appropriate LU 6.2 calls to VTAM. Sockets over SNA issues these LU 6.2 calls to VTAM using APPCCMD macroinstructions. Figure 2 on page 2 illustrates the relationships between application program, Sockets over SNA, and LU 6.2 interface components on a z/OS network node.

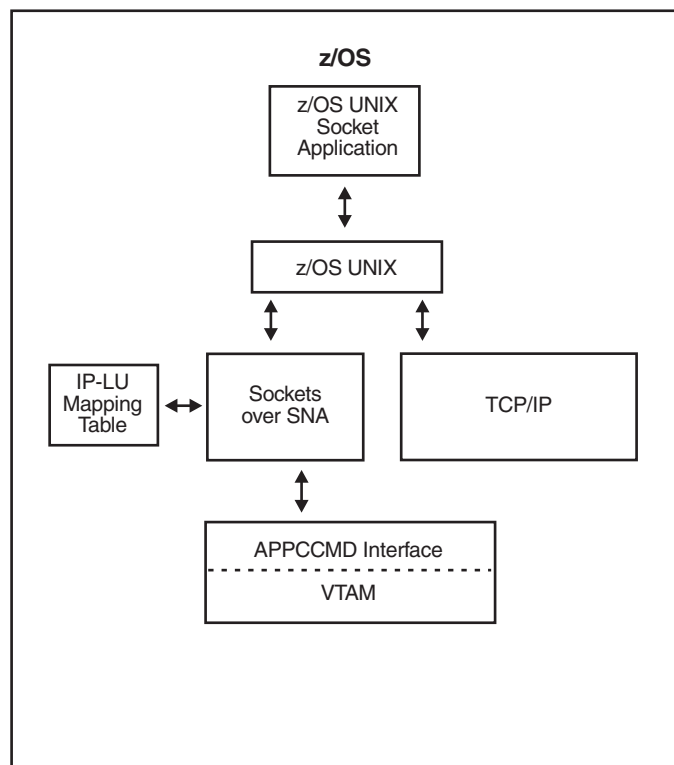


Figure 2. Structure of an AnyNet Sockets over SNA Node

Generating LU 6.2 Calls from Socket Calls

To enable sockets-formatted information to route over SNA, Sockets over SNA maps IP addresses to SNA fully qualified LU names. When an application program invokes Sockets over SNA to establish a stream connection with another application program, Sockets over SNA establishes one or two LU 6.2 conversations for the stream connection. If the nodes on which the communicating applications reside are both full-duplex enabled, only one conversation is established. If either of the communicating nodes is only half-duplex enabled, two conversations are established. The conversations are deallocated when the stream socket connection is ended. For more information about VTAM's LU 6.2 full-duplex support, refer to *z/OS Communications Server: SNA Programmers LU 6.2 Guide*. See "Requirements for Full-duplex Enablement" on page 6 for full-duplex requirements.

Sockets over SNA establishes one LU 6.2 conversation for all datagrams sent to a single destination. Conversations dedicated to datagram traffic are deallocated if they are unused for some specified period of time.

Mapping an IP Address to a Fully Qualified LU Name

When an application program invokes Sockets over SNA to communicate with another application program, it supplies the IP address of the destination node. Sockets over SNA must map the IP address to an SNA address to issue an appropriate LU 6.2 call. For every IP address that identifies a node, there will be a corresponding SNA fully qualified LU name.

See “Chapter 2. Planning for AnyNet Sockets over SNA” on page 7 for information about how address mapping works and for guidelines and requirements for setting up IP-LU address mapping.

Example Network Scenarios that Include Sockets over SNA

The following sample scenarios illustrate how nodes configured with Sockets over SNA can communicate in SNA networks. These samples are intended only to demonstrate how Sockets over SNA fits into SNA network scenarios; Sockets over SNA is compatible with all the system software levels listed in “System Software Required to Use AnyNet Sockets over SNA” on page 5.

Communicating from VTAM Subarea to VTAM Subarea

Figure 3 illustrates an SNA network allowing Socket Application A to communicate with Socket Application B through VTAM subarea attachments.

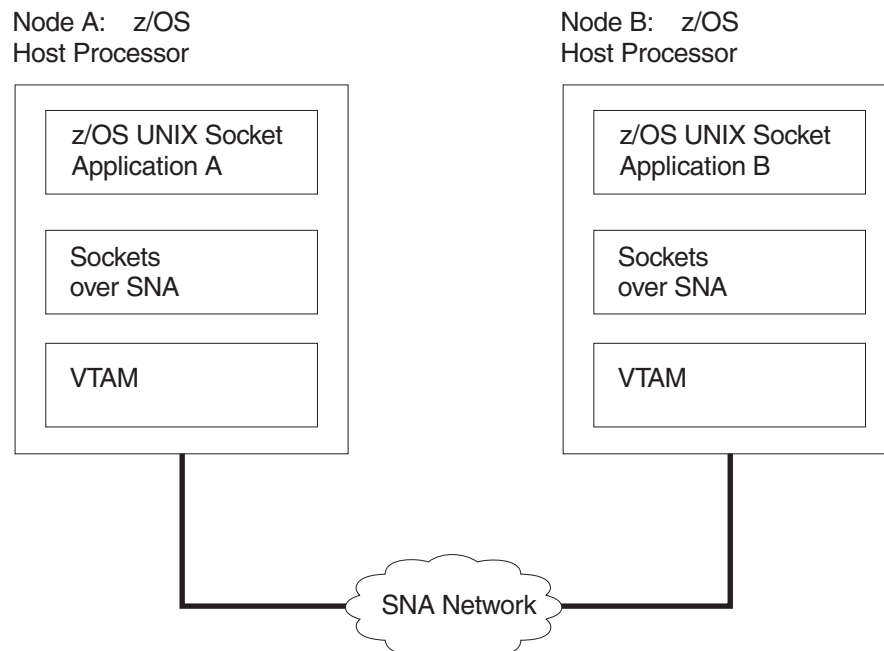


Figure 3. z/OS UNIX Socket Applications Communicating from VTAM Subarea to VTAM Subarea

Communicating from VTAM Subarea to OS/2 APPC through a 3172 Interconnect Controller

Figure 4 on page 4 illustrates an SNA network allowing Sockets Application A on a VTAM subarea node to communicate with Socket Application B through an IBM 3172 Interconnect Controller connected to a token ring.

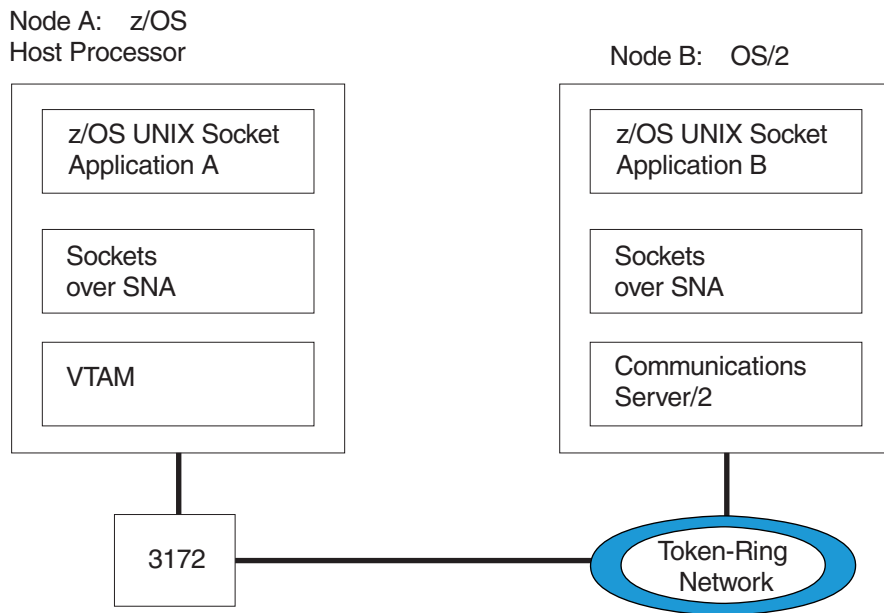


Figure 4. Socket Applications Communicating from VTAM Subarea to OS/2 APPC through a 3172 Interconnect Controller

Communicating from VTAM Subarea to OS/2 APPC through NCP/NTRI

Figure 5 on page 5 illustrates an SNA network allowing Application A to communicate with Application B through an NCP configured with NCP Token-Ring interconnection (NTRI).

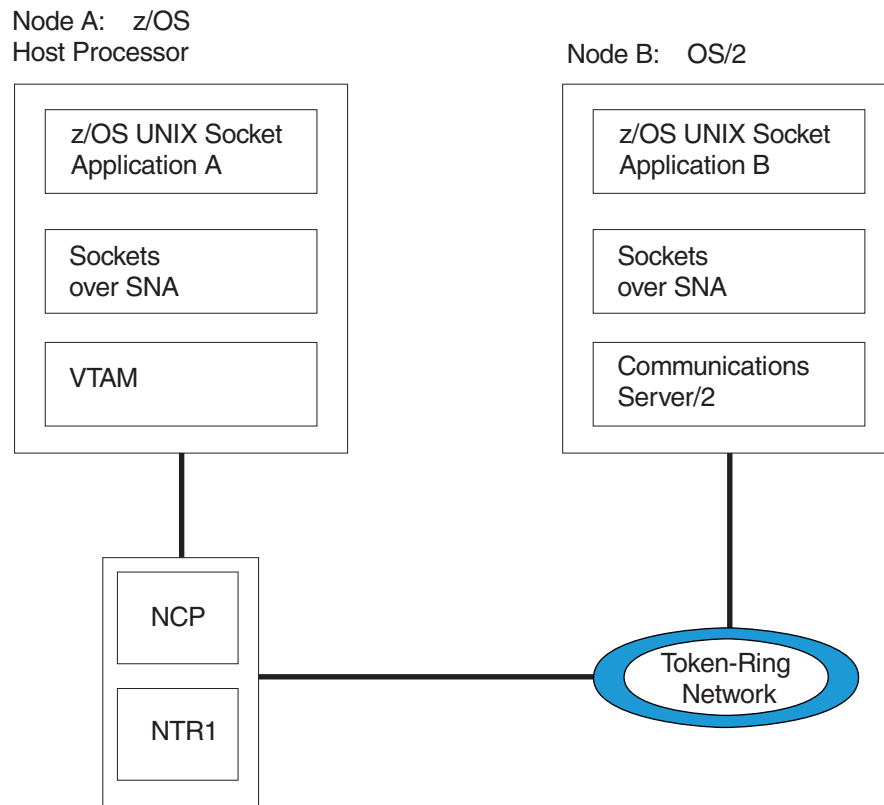


Figure 5. Socket Applications Communicating from VTAM Subarea to OS/2 APPC through NCP/NTRI

System Software Required to Use AnyNet Sockets over SNA

OS/390 V1R6 with Language Environment/370 (LE/370) is required to use AnyNet Sockets over SNA.

Applications Supported by AnyNet Sockets over SNA

Earlier releases of Sockets over SNA supported IBM TCP/IP for MVS/ESA applications, and also provided a BSD 4.3 C socket API for writing programs. These APIs are no longer supported by Sockets over SNA. Any existing applications written to these interfaces must be ported to z/OS UNIX System Services.

See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or *z/OS C/C++ Run-Time Library Reference* for information on socket calls offered by z/OS UNIX and for information on writing z/OS UNIX application programs. You can find information on porting applications to z/OS UNIX at the following URL:

<http://www.ibm.com/servers/eserver/zseries/zos/unix/index.html>

z/OS UNIX System Services Socket Applications Supported by Sockets over SNA

Sockets over SNA supports z/OS UNIX socket application programs using the AF_INET address family.

Some of the z/OS UNIX socket applications that Sockets over SNA supports are:

- FTP Server
- Web Server
- Telnet Server
- oping
- Rlogin

Requirements for Full-duplex Enablement

For a z/OS node to communicate in full-duplex mode, both connection partners must support full-duplex conversations. z/OS Communications Server and AnyNet Sockets over SNA support full-duplex conversations.

Managing Sockets over SNA

This section lists the network management tasks you can perform using the NetView program in an SNA network that includes Sockets over SNA. Because Sockets over SNA operates as an LU 6.2 application that uses SNA transport, you can use NetView and the NetView family of products to manage Sockets-over-SNA network activity. NetView Version 2 Release 1 (or later) provides network-management functions that allow you to:

- Display topology and status for SNA and TCP/IP devices which have been added to the resource object data manager (RODM)
- Conduct fault management using network alerts
- Trace session routes to identify failing nodes

Chapter 2. Planning for AnyNet Sockets over SNA

This chapter describes what you should consider before installing Sockets over SNA on z/OS. To use the information in this chapter, you should be familiar with the tasks associated with setting up a TCP/IP network and with defining SNA resources.

In a Sockets-over-SNA network, each Sockets over SNA node is identified by both an Internet Protocol (IP) address and an SNA address. Socket applications identify partner applications using IP addresses and port numbers. A socket application program uses IP addresses to tell Sockets over SNA which remote Sockets over SNA is being used by its partner application. Sockets over SNA, however, uses SNA addresses to specify source and destination locations because it operates as an APPC (LU 6.2) application program. Because a sockets application has no knowledge of SNA addressing, Sockets over SNA maps IP addresses provided by socket applications to SNA addresses in order to use the SNA transport protocol. The port number is used by the remote Sockets over SNA to identify one of its socket applications.

The SNA address refers to a network-qualified LU name.

When setting up Sockets over SNA, you must ensure that:

- An IP address is assigned to each Sockets over SNA node in the Sockets-over-SNA network
- Every IP address in the Sockets-over-SNA network is mapped to a unique SNA address (netid.luname)

Assigning IP Addresses to Sockets-over-SNA Nodes

This section describes considerations for assigning IP addresses in the Sockets-over-SNA network.

Assigning IP addresses to nodes in a Sockets-over-SNA network is the same as assigning IP addresses to nodes in a TCP/IP network. If you intend to connect the Sockets-over-SNA network to the Internet using a Sockets-over-SNA Gateway, you must first obtain one or more IP addresses. You may obtain an IP address from an Internet Service Provider (ISP) or by contacting the Network Information Center (InterNIC). The InterNIC assigns only the network portion of an IP address; your organization is responsible for assigning the host address portion.

You can access InterNIC services and information on the Internet at <http://www.internic.net/>. Select "Registration Services" to view registration options, procedures, and information.

You can also contact the InterNIC at:
Network Solutions
Attn: InterNIC Registration Services
505 Huntmar Park Drive
Herndon, VA 22070
1-703-742-4777

If both TCP/IP and Sockets over SNA are installed on the same node, the node must have two IP addresses because Sockets over SNA and TCP/IP operate with no knowledge of each other. One IP address identifies the node's connection to the TCP/IP network, the other IP address identifies the node's connection to the

Sockets-over-SNA network. See “Assigning an IP Address to the Sockets-over-SNA Interface” on page 12 for information on how routing decisions are made on a z/OS node.

If you intend to use both TCP/IP and Sockets over SNA to connect the same network of nodes, you must do either of the following:

- Create one or more different subnetworks (subnets) for all nodes using Sockets over SNA. If you want to use an existing NIC-registered IP address, you can set up a subnetwork for all Sockets-over-SNA nodes. If you have OS/2 machines in your network that use both AnyNet/2 Sockets over SNA and OS/2 TCP/IP, be aware that these products do not support variable subnetworking.
- Use different IP addresses altogether to distinguish Sockets-over-SNA nodes from TCP/IP nodes.

Notes:

1. You can only use one instance of Sockets over SNA for each z/OS host.
2. If you are using common INET, you can use TCP/IP and Sockets over SNA together. z/OS UNIX routes information over either Sockets over SNA or TCP/IP as appropriate, according to the configuration of each.

If you are not using common INET, either TCP/IP or Sockets over SNA can be the z/OS UNIX transport provider, but not both.

Assigning SNA Addresses to Sockets-over-SNA Nodes

This section describes considerations for assigning SNA addresses to Sockets-over-SNA nodes. Read this section and “Mapping IP Addresses to SNA Addresses” on page 9 before you start to configure Sockets over SNA.

In addition to determining which IP addresses to use, you have to determine which SNA addresses to assign to access nodes in the network. For Sockets over SNA, you need to determine the following:

- Name of the SNA network in which the Sockets-over-SNA nodes reside
- SNA LU names that identify Sockets-over-SNA nodes

Assigning SNA Network Names to Sockets-over-SNA Nodes

To ensure every SNA network name is unique, it is recommended that you register the network name with the IBM SNA Network Registry. The SNA Network Registry serves as a central data base for registered SNA network names. If your SNA network connects to another SNA network, having registered both network names ensures their ability to successfully exchange information. To register an SNA network name, contact your IBM branch office representative.

Assigning SNA LU Names to Sockets-over-SNA Nodes

Before defining LU names for the Sockets-over-SNA network, it is recommended that you establish an LU naming convention. Using a predefined naming convention for Sockets-over-SNA LUs helps you:

- Control which names are used for Sockets-over-SNA LUs and which names are used for other LUs in your SNA network.
- Analyze and resolve mapping and routing errors, should they occur.

For example, you might establish a convention that all LUs used by MPTN access nodes and Sockets over SNA begin with the letters “SX”.

The following sections describe two methods that you can use to assign LU names. Regardless of which method you use to generate LU names from IP addresses, you may need to define the LU names to VTAM. See “Defining Remote Nodes to Sockets over SNA” on page 17 for more information on defining remote nodes in a non-APPN-enabled network.

Method 1: Allow Sockets over SNA to Generate LU Names Algorithmically

This method is most effective for networks made up of more than a few nodes. Using algorithmically generated LU names simplifies the administration associated with IP-LU mapping and reduces the probability of defining erroneous addresses.

To generate LU names algorithmically, use the ISTSKMAP utility to do the following:

- Determine the IP address of the local node and the SNA network name and LU-name template used by all nodes in the network.
- Specify the address mask used for the Sockets-over-SNA network. Note that you do not specify an address mask with all bits set (do not use 255.255.255.255) when algorithmically defining LU names. See “Mapping IP Addresses to SNA Addresses” for a description of how the address mask is used in IP-LU mapping.

Individual mappings for remote destination nodes do not have to be defined on the local node. Sockets over SNA generates the entire LU name using the unmasked bits of the IP address and the LU name template. For example, each node in the network can define “SX” as the LU template. For each node, Sockets over SNA generates the remaining 6 characters of the LU name using the bits of the IP address that correspond to “0” bits in the address mask.

Method 2: Define an LU Name for Each Node in the Network

This method is most effective if a very small number of nodes are using Sockets over SNA, or if you are initially setting up the network and want to test communication among a few nodes. Be aware, however, that assigning LU names to individual nodes creates considerable administrative overhead.

To define each node in the network, use the ISTSKMAP utility to do the following:

- Determine the IP address, SNA network name, and LU name of the local node.
- Specify an address mask of 255.255.255.255 for the local node. Defining an address mask with all bits set (255.255.255.255) creates an explicit mapping of the specified IP address to the specified SNA network name and LU name.
- At each remote node in the network, define the IP address, SNA network name, and LU name of the node. Specify an address mask of 255.255.255.255.

For example, to define a network of 20 nodes, where each node can communicate with the other 19 nodes, use the following steps:

1. At each node define the IP address of the local node.
2. Define the IP address and LU name of all 19 remote nodes.

If after initial configuration you add a node to the network, you must ensure that all nodes that may communicate with the new node locally define the IP address and LU name of the new node.

Mapping IP Addresses to SNA Addresses

This section describes how an IP address is mapped to an SNA address.

Generating an SNA Address from an IP Address

The address mapping process has two steps:

1. The first portion of the IP address is mapped to an SNA network name and an LU-name template.
2. The rest of the IP address is used to generate the remainder of the LU name not defined by the LU-name template. This step is algorithmic: Sockets over SNA generates the LU name using only the second portion of the IP address and the LU-name template.

Step 1: Mapping an IP Network ID to an SNA Network Name and LU-Name Template

In this step, Sockets over SNA uses the IP-LU mapping table to determine which SNA network name and LU-name template to use in mapping a given destination address. Each line entry in the table has the following four fields:

- IP address
- Address mask
- SNA network name
- LU-name template

The first two fields, IP address and address mask, are used by Sockets over SNA to determine whether a given destination address matches the IP-LU mapping table entry.

IP address

When Sockets over SNA determines that a destination address is reachable via the SNA network, it searches the IP-LU mapping table for an entry that matches the destination address. Sockets over SNA uses the address mask to determine which bits of the destination address should be compared with the IP address in an IP-LU mapping table entry.

Address mask

A “1” bit in the mask indicates that the corresponding bits in the destination address and the IP address in the IP-LU mapping table entry must match. Bits in the destination address that correspond to “0” bits in the address mask will be used later to generate the LU name. For example, if the address mask sets the first 24 bits to “1”, when the first 24 bits of the destination address match the first 24 bits of an IP address in the IP-LU mapping table, the remaining 8 bits are used to generate the LU name.

The last two fields of a line entry in the mapping table, SNA network name and LU-name template, are used to generate the SNA network address.

SNA network name

When Sockets over SNA selects an entry in the IP-LU mapping table, it maps the masked bits of the IP address entry to the SNA network name.

LU-name template

Sockets over SNA uses the characters in this field as the fixed positional characters of the LU name (LU-name template) corresponding to the unmasked bits of the IP address. The LU-name template enables you to define an LU-name prefix that:

- Ensures that the LU name begins with a valid character
- Distinguishes LUs used for Sockets-over-SNA communication from other LUs that communicate over an SNA network

Step 2: Generating the LU Name

The algorithmic mapping step generates the characters that are not specified in the LU-name template and combines these characters with the LU-name template to form the 8-character LU name. The portion of the destination address not masked by the address mask is used to generate the portion of the LU name you do not specify in the LU-name template.

If you specify an IP-LU entry with an address mask with all bits set (255.255.255.255), you indicate that you are not using algorithmic mapping. In this case, only the characters you specify in the LU-name template are used to generate the LU name; no additional LU characters are generated. See “Assigning SNA LU Names to Sockets-over-SNA Nodes” on page 8 for information on when to explicitly map individual IP addresses.

Example of an Entry in the IP-LU Mapping Table

Refer to the sample mapping table shown in Table 1. The entry in this table indicates that IP addresses whose upper 24 bits (255.255.255.0) have the value 128.109.130 should be mapped to the SNA network USCOMDIV. The LU name template defines SX as the first two positional characters for the LU-name prefix; the remaining six characters of the LU name are generated algorithmically from the last 8 bits of the IP address. For example, if Sockets over SNA receives the IP address 128.109.130.45 and processes this address using the entry in Table 1, Sockets over SNA produces an SNA address of USCOMDIV.SX00001D.

Table 1. Example of an Entry in the IP-LU Mapping Table

IP Address	Address Mask	SNA Network Name	
		LU-Name Template	
128.109.130.0	255.255.255.0	USCOMDIV	SX

Note: You are responsible for setting up unique mappings; Sockets over SNA does not ensure that the LU names you define are unique to the Sockets-over-SNA network. In Table 1, you could have added:

128.109.140.0 255.255.255.0 USCOMDIV SX

In this case, two different IP addresses, 128.109.130.123 and 128.109.140.123, would map to the same SNA network name and LU name.

How Sockets over SNA Uses IP-LU Mapping Table Entries

Each time a Sockets-over-SNA node is started, at least one entry in the IP-LU mapping table must be defined. Depending on how your network is configured, you might need to define more than one entry.

Following is an example of how Sockets over SNA would use two IP-LU mapping table entries when it is first started. Assume that the entries have been defined using the ISTSKMAP utility:

```
ISTSKMAP add 128.109.139.0 255.255.224.0 USIBMCO SX0
ISTSKMAP add 128.109.0.0 255.255.0.0 USCOMDIV SX1
```

The first entry tells Sockets over SNA to map all IP addresses that match 128.109.139.0 in the first 19 bits to the SNA network name USIBMCO; the remaining 13 bits are used to generate the LU name that begins with SX0. The second entry tells Sockets over SNA to map all IP addresses that match 128.109.0.0 in the first 16 bits to the SNA network name USCOMDIV; the remaining 16 bits are used to generate the LU name that begins with SX1.

All addresses that qualify for the first map (that is, addresses that begin with 128.109.139) also qualify for the second map (128.109). Sockets over SNA first searches the mapping table for the longest address mask, where “longest” means the mask with the most bits set. If the address supplied to Sockets over SNA does not match the address generated using the longest mask, the next longest mask is tried. The process repeats until a match is found, or until there are no more entries in the mapping table. In this case, the first entry is tried first; if Sockets over SNA receives an IP address that does not match the first entry, Sockets over SNA determines whether the address could be mapped by the second entry.

Summary of the IP-LU Mapping Process

Following is the process that maps an IP address to an SNA address:

1. Sockets over SNA receives an IP address for a destination that can be reached using the SNA network.
2. Sockets over SNA queries the IP-LU mapping table to determine the following:
 - What portion of the IP address to map to an SNA network name
 - Which SNA network name maps to the masked portion of the IP address
 - Which LU-name template is used in calculating the LU name
 - What portion of the IP address to use to calculate the remainder of the LU name
3. Using the information supplied in the IP-LU mapping table, Sockets over SNA passes the SNA network name and LU name of the destination host to VTAM.

Assigning an IP Address to the Sockets-over-SNA Interface

When Sockets over SNA is started, an IP address must be assigned to the network interface (sna0). Assigning an IP address to the sna0 interface activates the interface of the local host. VTAM and the APPL statement for Sockets over SNA must be active in order for the sna0 interface to be activated. Sockets over SNA starts monitoring sna0 for information destined for this node.

The following steps determine whether data received by Sockets over SNA is routed through the Sockets-over-SNA interface:

1. Sockets over SNA receives an IP address.
2. Sockets over SNA searches in the routing table to determine which interface is used to route data for the given IP address.
3. If the interface chosen is the Sockets-over-SNA (sna0) interface, Sockets over SNA uses the mapping information supplied during configuration to generate an SNA address. See “Mapping IP Addresses to SNA Addresses” on page 9 for information on mapping IP addresses to SNA addresses.
4. If the interface chosen is the local loopback interface (1o) instead, Sockets over SNA routes the request back to itself.
5. If no interface is found, Sockets over SNA cannot route the data; the API call of the application program fails and returns completion error code ENETUNREACH or EHOSTUNREACH.

The address mask specified using ISTSKIFC determines which remote IP addresses can be reached directly through the sna0 interface. (The ISTSKRTE utility is used to determine which remote IP address can be reached indirectly through the sna0 interface. See “Using the ISTSKRTE Utility to Update the Local Sockets over SNA Routing Table” on page 23 for information on the ISTSKRTE utility.) Because Sockets over SNA must be able to map its own IP address to an LU name when the

address is assigned to `sna0`, an `ISTSKMAP` utility must execute before the `ISTSKIFC` utility executes. The `ISTSKMAP` utility establishes the IP-LU address mapping for the local node so the information can be used when the `ISTSKIFC` utility assigns the address to the Sockets-over-SNA interface. See “Starting and Initializing Sockets over SNA” on page 35 for an example of the setup JCL used to start Sockets over SNA.

The mask defined using `ISTSKMAP` and the mask defined using `ISTSKIFC` perform different functions. The `ISTSKIFC` mask is used to determine which remote IP addresses can be reached directly through the `sna0` interface; the `ISTSKMAP` mask determines what portion of a given IP address is mapped to an SNA network name.

Domain Name Server

In previous releases, Sockets over SNA provided a name query function for use with a domain name server. It is no longer possible to define `HOSTS`, `NETWORKS`, `PROTOCOL`, `RESOLV`, and `SERVICES` files to Sockets over SNA. Instead, z/OS UNIX applications get this information from z/OS UNIX. You must first provide this information to z/OS UNIX. Refer to *z/OS UNIX System Services Planning* for a description of how to define these services to z/OS UNIX.

Defining ETC Hosts, Networks, Protocols, and Services to z/OS UNIX System Services

Files formerly defined to Sockets over SNA are now defined to z/OS UNIX System Services. The files that must be defined to z/OS UNIX are `/etc/hosts`, `/etc/networks`, `/etc/protocol` and `/etc/services`.

Although we identify these files here with HFS style names that suggest BSD formatted files, you are not restricted to HFS files nor to BSD format when defining these files to z/OS UNIX.

If you plan to use both Sockets over SNA and TCP/IP with z/OS UNIX, you should define ETC files common to Sockets over SNA and TCP/IP.

z/OS UNIX System Services Planning describes these options, as well as other options, to define ETC files to z/OS UNIX System Services.

Chapter 3. Defining AnyNet Sockets over SNA

This chapter describes the procedures used to define Sockets over SNA to VTAM. Make sure that you have finished the installation procedures described in the *z/OS Program Directory* before you attempt the procedures in this chapter. The program directory describes the JCL instructions for transferring all Sockets-over-SNA files from tape to host storage.

Defining AnyNet Sockets over SNA to VTAM

This section describes the procedures you follow to define z/OS nodes to use Sockets over SNA. To perform the tasks described in this section, you should be familiar with defining an application and an independent LU to VTAM, and be aware of the following:

- Sockets over SNA running locally on a z/OS node operates as an APF-authorized VTAM LU 6.2 application.
- You must define at least one MODEENT macro entry for all sessions established for Sockets over SNA.
- All destination nodes to which connections will be established must be defined as independent LUs on the local VTAM host. The independent LUs can be statically defined in a VTAMLST member or you can allow VTAM to dynamically define the independent LUs.

Defining APPL Statements for Sockets over SNA

All rules and guidelines for defining VTAM APPL definition statements apply to defining Sockets over SNA to VTAM. Sockets over SNA operates as an LU 6.2 application. The simplest way to define the application is to use VTAM's dynamic definition of VTAM applications support.

The following is a sample source APPL definition for Sockets over SNA:

```
SX          VBUILD TYPE=APPL
SX*         APPC=YES,
            PARSESS=YES,
            DSESLIM=10,
            DMINWNL=5,
            DMINWNR=5,
            AUTOSESS=0,
            AUTH=(ACQ,PASS),
            OPERCNOS=ALLOW,
            ATNLOSS=ALL
```

The above example assumes that your LU template is "SX".

You may also define the application by determining the APPL name that will be used by the local Sockets over SNA application. Use the ISTSKMAP utility with the convert option to determine the application name based on the IP address of the local Sockets over SNA.

Refer to the *z/OS Communications Server: SNA Resource Definition Reference* for information on defining APPL statements.

Increasing the Session Limit for Sockets over SNA

This section describes how to use the DSESLIM operand to increase the upper limit for the number of sessions used by Sockets over SNA.

Each stream socket connection that is established uses two half-duplex LU 6.2 conversations (one for each direction) or one full-duplex conversation. Datagram connections use half-duplex LU 6.2 conversations as needed. Depending on the number of concurrent Sockets-over-SNA stream and datagram connections used by your installation, you might need to increase the upper limit for the number of sessions used by Sockets over SNA. The DSESLIM operand defines a session limit. Refer to the *z/OS Communications Server: SNA Resource Definition Reference* for information on the DSESLIM operand.

Defining a MODEENT Macro Entry for Sockets over SNA

When Sockets over SNA establishes a conversation with a remote Sockets over SNA, it must supply a mode name to VTAM. VTAM uses the mode name to identify characteristics of the connection between the two Sockets-over-SNA nodes.

By default, Sockets over SNA uses a mode name of SNACKETS for all conversations it initiates. If you want all conversations initiated by a particular Sockets over SNA to use a different mode name, specify this mode name in the SXMODE_DEFAULT environment variable:

```
SXMODE_DEFAULT = DFLTMODE
```

See “Defining Sockets-over-SNA Environment Variables: the ENVVAR Data Set” on page 27 for more information on defining Sockets-over-SNA environment variables.

Defining Mode Names for Specific Ports

For stream socket connections, you can specify a mode name based on the remote port number specified by the application in its connect() API call. For example, assume you are running batch application Z whose server waits for connections on port 5678. To make all connections from Z clients running on this Sockets over SNA use mode name BATCHMOD, you would define an environment variable:

```
SXMODE5678 = BATCHMOD
```

For most applications, the SXMODE n entry needs to be defined on the client node because the client typically initiates the connection to the server.

You can specify as many SXMODE n variables as you need. You can include both an SXMODE_DEFAULT and SXMODE n variables in ENVVAR. Connections to any port number that does not have a corresponding SXMODE n variable use the mode name defined by the SXMODE_DEFAULT variable.

Conversations established for datagram sockets always use the default mode name (either SNACKETS or the value you specify for SXMODE_DEFAULT).

See “SXMODE_DEFAULT” on page 31 and “SXMODE n ” on page 31 for more information on defining these environment variables.

Specifying an Alternate Logon Mode Table

Because of the way VTAM handles mode names, **all** mode names that are used by Sockets over SNA must be defined in a single logon mode table. Unless you specify otherwise, VTAM attempts to find all the modes requested by Sockets over SNA in the default logon mode table, ISTINCLM. To make VTAM search another logon mode table for the mode names, specify the name of the table using the MODETAB keyword on the APPL definition statement for Sockets over SNA.

Following is a sample logon mode table:

```

MYTABLE  MODETAB
*****
*
* You might recognize these as the entries for #INTER and #BATCH *
* from ISTINCLM.
*
*****
DFTLMODE  MODEENT  LOGMODE=DFTLMODE,FMPROF=X'13',TSPROF=X'07',
           ENCR=B'0000',SSNDPAC=7,RUSIZES=X'F7F7',
           SRCVPAC=7,PSNDPAC=7,APPNCOS=#INTER
BATCHMOD  MODEENT  LOGMODE=BATCHMOD,FMPROF=X'13',TSPROF=X'07',
           ENCR=B'0000',SSNDPAC=3,RUSIZES=X'F7F7',
           SRCVPAC=3,PSNDPAC=3,APPNCOS=#BATCH
MODEEND

```

See the *z/OS Communications Server: SNA Resource Definition Reference* for complete instructions on coding and using your own logon mode table.

Defining Remote Nodes to Sockets over SNA

Sockets over SNA can communicate with other VTAM hosts and workstations that implement Sockets over SNA. Remote workstations typically have to be defined as independent LUs (ILU) in order for the local Sockets over SNA to establish sessions with them. A remote ILU is defined to VTAM as a cross-domain resource (CDRSC). The CDRSC may be predefined or you can allow VTAM to dynamically create the CDRSC. The session path may be any type supported by the remote node including subarea, APPN or LEN connections. Refer to the *z/OS Communications Server: SNA Network Implementation Guide* for information on defining independent LUs as CDRSC.

If a remote socket application initiates communication with a socket application in a VTAM subarea, VTAM can dynamically define the LU associated with the remote Sockets over SNA. Refer to the *z/OS Communications Server: SNA Network Implementation Guide* for information on dynamically defining remote LUs and for examples of CDRSC definition statements.

If you choose to predefine ILUs or LUs, you have to determine which LU names are generated from the IP addresses you are using. Use the following steps to determine and define the LU names:

- List the IP addresses of all nodes that have to be defined to the local VTAM.
- Use the convert option of the ISTSKMAP utility and specify a range of IP addresses that have to be mapped to LU names.
- If there are a large number of nodes involved, redirect the output from the ISTSKMAP convert command to a file.

The following example shows how ISTSKMAP can be called under TSO to produce LU names corresponding to the range of IP addresses specified on the convert option. The ISTSKMAP utility resides in the ISTSKMAP member of your VTAMLIB, for example, SYS1.VTAMLIB. Output is redirected to a data set called USERN.OUTPUT.

```

===> call 'sys1.vtamlib(istskmap)' 'convert 128.109.140.1
128.109.140.5 0xfffffc00 samp >'usern.output'

```

The previous call produces the following output:

```
> ISTSKMAP convert 128.109.140.1 128.109.140.5 0xffffffff00 samp
128.109.140.1 SAMP0001
128.109.140.2 SAMP0002
128.109.140.3 SAMP0003
128.109.140.4 SAMP0004
128.109.140.5 SAMP0005
```

- Using the listing of LU names from the ISTSKMAP convert operation, define to VTAM the appropriate independent LU statements for each remote Sockets over SNA to which a conversation is initiated from the local Sockets over SNA.

The ISTSKMAP utility is a z/OS UNIX application; refer to the *z/OS Language Environment Programming Reference* for information on running z/OS UNIX applications and on redirecting output under z/OS. See “Using the ISTSKMAP Utility to Establish, Update, and Query the Mapping Table” for directions on using the ISTSKMAP convert option.

Using the ISTSKMAP Utility to Establish, Update, and Query the Mapping Table

This section describes:

- How to use the ISTSKMAP utility to add, convert, delete, update, and query entries in the IP-LU mapping table
- How to use the ISTSKMAP convert option to generate a listing of LU names corresponding to a range of IP addresses

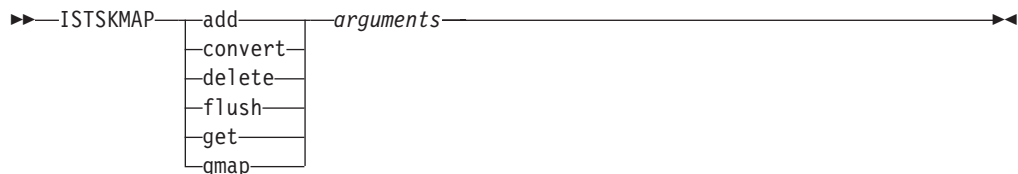
The ISTSKMAP utility can be run either interactively (for example, using TSO), or using batch (for example, JCL). To run ISTSKMAP interactively from TSO, the z/OS UNIX shell, or from batch, your logon procedure must be set up to run authorized LE/370 application programs. Refer to your *z/OS Language Environment Programming Reference* for information on setting up your logon procedure to run LE/370 application programs.

Following is an example of how to run ISTSKMAP in TSO with the interface and IP address arguments:

```
call 'sys1.vtamlib(istskmap)' 'add 128.109.140.0 255.255.224.0 USIBMNR SX'
```

Syntax of the ISTSKMAP Utility

The syntax for the ISTSKMAP command follows. See “Appendix A. Syntax Conventions” on page 47 for information on how to interpret the syntax diagrams for the utilities described in this book.



add

adds a new line entry to the IP-LU mapping table. To replace an existing entry with a new entry that has the same IP address, delete the existing entry first, and then add the new entry.

convert

displays the LU name that is generated from the specified IP address(es), network mask, and LU-name template. The **convert** operand does not use the IP-LU mapping table.

See “Using the Convert Option to Generate LU Names” on page 21 for information on using the convert option.

delete

removes the specified entry from the IP-LU mapping table.

flush

removes all entries from the IP-LU mapping table.

CAUTION:

Be sure you want to erase all entries in the table before you issue this command.

get

displays the contents of the IP-LU mapping table.

qmap

displays the current LU-name mapping for the specified IP address. The **qmap** operand accesses the IP-LU mapping table to display the LU name, but does not make any changes to the table.

arguments

select arguments depending on the operand you use.

add

ip_address
addr_mask
sna_netname
lu_template

convert

ip_address
addr_mask
lu_template

convert

start_ip
end_ip
addr_mask
lu_template

flush none

qmap *ip_address*

get none

Following is a description of each argument:

ip_address

specifies the IP address. Use either hexadecimal notation or dotted-decimal notation.

Hexadecimal notation is in the form 0xaabbccdd, where:

- 0x is a required prefix indicating that the value is in hexadecimal format

- Each portion of the address (*aa*, *bb*, *cc*, or *dd*)
 - Is a two-digit hexadecimal value ranging from 00 to FF
 - Represents 8 bits of a 32-bit IP address

Dotted-decimal notation is in the form *a.b.c.d*, where each portion of the address (*a*, *b*, *c*, or *d*):

- Is a decimal value ranging from 0 to 255
- Is delimited by a period
- Represents 8 bits of a 32-bit IP address

Specify all four 8-bit sections. For example, these are valid address entries:

```
190.12.0.1
50.218.29.0
129.67.0.0
```

However, the value 201.0 would not be valid because it does not consist of four sections.

If you are mapping an IP address for a single host to one SNA network name and LU name, specify the entire IP address. Otherwise, specify only the IP network ID and use a host ID of 0.

start_ip

specifies the beginning IP address for a range of addresses. Use beginning and ending arguments when you want to generate corresponding LU names for a range of IP addresses. All rules listed for *ip_address* apply.

end_ip

specifies the ending IP address for a range of addresses. Use beginning and ending arguments when you want to generate corresponding LU names for a range IP addresses. All rules listed for *ip_address* apply.

addr_mask

specifies which bits in a destination IP address are compared to the IP address in an IP-LU mapping table entry. The unmasked bits are used to generate the LU name. Use either hexadecimal notation or dotted-decimal notation. When using hexadecimal notation, precede the value with 0x (for example, 0xFFFFFFFF00).

If you are mapping an IP address for a single host to one SNA network name and LU name, mask the entire address with either 0xFFFFFFFF or 255.255.255.255.

sna_netname

specifies the SNA network name. The same restrictions that apply to naming an SNA network name apply here. Specify from 1 to 8 characters.

lu_template

specifies the LU-name template. The same restrictions that apply to naming an SNA LU apply to an LU-name template.

Note: The character you specify in the first character position has to be a valid LU-name character to generate a valid SNA network address. Valid first characters for LU names are A–Z, @, #, and \$.

The number of characters you can specify ranges from 1 to 8, depending on the size of the mask. Table 2 on page 21 describes the range of characters you can specify for the LU template.

The ISTSKMAP utility reports an error if you specify more characters in the LU template than are allowed, or if the LU template does not begin with a valid character. To define a specific LU name, specify a 32-bit address mask (for example, 0xFFFFFFFF).

Table 2. Limitations for Defining LU Templates

Number of Bits in the Address Mask:	Range of Address Mask:	Size of LU template:
8–11 (includes class A addresses)	0xFF000000–0xFFE00000	1–3 characters
12–16 (includes class B addresses)	0xFFFF0000–0xFFFFF000	1–4 characters
17–21	0xFFFFF8000–0xFFFFF800	1–5 characters
22–26 (includes class C addresses)	0xFFFFF000–0xFFFFF000	1–6 characters
27–31	0xFFFFF000–0xFFFFF000	1–7 characters
32	0xFFFFFFFF	1–8 characters

If you specify all 8 characters, you effectively define a specific LU name and SNA network name corresponding to the IP address. You must also specify a complete IP address and an address mask of either 0xFFFFFFFF or 255.255.255.255.

Always use dots (periods) to indicate unspecified characters. Characters do not have to be contiguous; for example, the template N..R.P is valid. You do not have to indicate trailing character positions. For example, if you specify SX as the first two characters in the template, no periods are required. If you specify a template with characters that are positioned at the end of the LU name, use periods to indicate all positions up to the specified characters (for example, “N.....RP”).

Using the Convert Option to Generate LU Names

The ISTSKMAP convert option is useful for defining SNA destination LUs on low-entry networking (LEN) nodes. Your network might contain LEN nodes that must be locally defined to VTAM. If so, first determine the range of IP addresses you want to use to identify the remote nodes, then use convert to generate a list of LU names corresponding to the IP addresses.

Following is an example of how to generate the corresponding LU name for one IP address:

```
ISTSKMAP convert 128.109.139.4 255.255.0.0 SX
```

The previous command returns:

```
128.109.139.4 maps to LU name SX0012R4
```

Following is an example of how to generate the corresponding LU names for a given range of IP addresses:

```
ISTSKMAP convert 128.109.139.4 128.109.139.7 255.255.0.0 SX
```

The previous command returns:

```
128.109.139.4 SX0012R4
128.109.139.5 SX0012R5
128.109.139.6 SX0012R6
128.109.139.7 SX0012R7
```

Using the ISTSKIFC Utility to Define an SNA Network Interface for Sockets over SNA

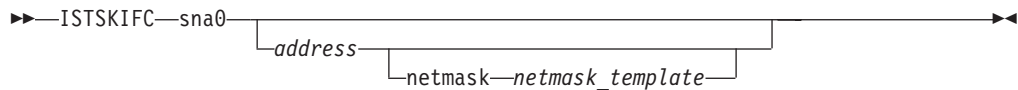
This section describes how to use the ISTSKIFC utility to assign an IP address to the SNA network interface used by Sockets over SNA. The name of the interface to the SNA network for Sockets over SNA is hard-coded as the value `sna0`. An IP address must be assigned to `sna0` before applications try to establish connections.

The ISTSKIFC utility can be run either interactively (for example, using TSO), or using batch (for example, JCL). To run ISTSKIFC interactively from TSO, the z/OS UNIX shell, or from batch, your logon procedure must be set up to run authorized LE/370 application programs. Refer to your *z/OS Language Environment Programming Reference* for information on setting up your logon procedure to run LE/370 application programs.

Following is an example of how to run ISTSKIFC in TSO with the interface and IP address arguments:

```
call 'sys1.vtamlib(istskifc)' 'sna0 128.109.140.6'
```

The format of the ISTSKIFC utility is:



Note: The ISTSKIFC utility supports more options and operands than those described here. The only options described here are those that apply specifically to Sockets over SNA.

Following is a description of each operand.

sna0

specifies an interface to the SNA network.

If you specify only `sna0`, ISTSKIFC displays the status of the `sna0` interface. Status includes whether the interface is active (up) and the address assigned to it.

address

specifies the IP address to assign to the interface. Specify either a host name or an IP address:

Host name

The ISTSKIFC utility queries the host file and resolves the name into the corresponding IP address.

IP address

Use either hexadecimal notation or dotted-decimal notation.

Hexadecimal notation is in the form `0xaabbccdd`, where:

- `0x` is a required prefix indicating that the value is in hexadecimal format.

- Each portion of the address (*aa*, *bb*, *cc*, or *dd*)
 - Is a two-digit hexadecimal value ranging from 00 to FF
 - Represents 8 bits of a 32-bit IP address

Dotted-decimal notation is in the form *a.b.c.d*, where each portion of the address (*a*, *b*, *c*, or *d*):

- Is a decimal value ranging from 0 to 255
- Is delimited by a period
- Represents 8 bits of a 32-bit IP address.

Note: Do not use leading zeros to specify values in dotted-decimal notation, for example, 029.67.1.1. If you specify an integer value using one or more leading zeros, the value is interpreted as an octal value instead of a decimal value.

netmask *netmask_template*

The **netmask** keyword with the *netmask_template* specifies the number of bits in the IP address that will be used to define the network ID. This operand pair is optional. If you assign an IP address to the `sna0` interface, and you do not specify a netmask, Sockets over SNA determines the network ID portion of the address by reading the first three (uppermost) bits. The settings of the first 3 bits indicate the classification of the address.

Table 3 shows, in decimal notation, the address ranges corresponding to address classifications A, B, and C. For a given dotted-decimal IP address expressed in the form *aaa.bbb.ccc.ddd*, the range of values relate to the portion of the address represented by *aaa*.

Table 3. Determining the Classification of an IP Address

This range of values for <i>aaa</i> :	Indicates this address class:	And this netmask:
1–127	Class A	255.0.0.0
128–191	Class B	255.255.0.0
192–255	Class C	255.255.255.0

Use either hexadecimal notation or dotted-decimal notation. The same rules for defining the IP address apply to defining the network mask. When using hexadecimal notation, precede the value with 0x (for example, 0xFFFFFFFF00).

Following are some examples using `ISTSKIFC`:

```
ISTSKIFC sna0 xipe
ISTSKIFC sna0 128.109.140.1
ISTSKIFC sna0 128.109.140.1 netmask 255.255.0.0
```

Using the `ISTSKRTE` Utility to Update the Local Sockets over SNA Routing Table

This section describes how to use the `ISTSKRTE` utility to add and delete entries in the local routing table used by Sockets over SNA. If you are using the AnyNet/2 Sockets over SNA Gateway to route information to a native TCP/IP network, use the `ISTSKRTE` utility on z/OS to establish routes to all nodes that can be reached through the Sockets Gateway. (Refer to *IBM Communications Server for OS/2 Warp: Guide to AnyNet Sockets over SNA* for information on using Sockets Gateway.) You can also use the `ISTSKRTE` utility to delete routing table entries.

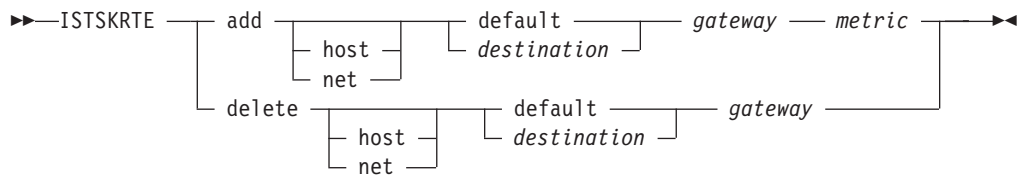
The ISTSKNST utility is used to display the contents of the Sockets-over-SNA routing table. See “Using the ISTSKNST Utility to Display Sockets-over-SNA Routing Tables” on page 40 for information on the ISTSKNST utility.

The ISTSKRTE utility can be run either interactively (for example, using TSO), or using batch (for example, JCL). To run ISTSKRTE interactively from TSO, the z/OS UNIX shell, or from batch, your logon procedure must be set up to run authorized LE/370 application programs. Refer to your *z/OS Language Environment Programming Guide* for information on setting up your logon procedure to run LE/370 application programs.

Following is an example of how to run ISTSKRTE in TSO with the -? for a quick view of the usage syntax:

```
call 'sys1.vtamlib(istskrte)' '-?'
```

The syntax for the ISTSKRTE utility follows:



Following is a description of each operand:

(-? or no operand)

If you specify the ISTSKRTE utility with a question mark, unrecognizable operand, or no operand, ISTSKRTE displays the following usage syntax:

Usage:

```
ISTSKRTE: [-?] (add | delete) (net | host) (<destination> | default)
           <gateway> [<metric>]
```

Where:

-?	Displays this message.
add	To add a route.
delete	To delete a route.
net	To add or delete a network.
host	To add or delete a host.
destination	Is the destination host or network.
default	All destinations not defined with another routing table entry.
gateway	Is the next-hop in the path to the destination.
metric	Number of hops to destination. Metric is required for add commands.

Example: ISTSKRTE add net 129.34.10.0 129.34.10.60 1

add

Adds a route entry to the local Sockets-over-SNA routing table. If you do not specify either the host or net keyword, host is assumed.

If you specify host, add adds an entry for the node specified as *destination*.

If you specify net, add adds an entry for the network specified as *destination*.

delete

Deletes the entry specified as *destination* from the local Sockets-over-SNA routing table.

host

Causes the specified *destination* to be interpreted as the address of a single node.

net

Causes the specified *destination* to be interpreted as a network address.

destination

Destination host or network for which you are defining a route entry. Specify either an IP address in dotted-decimal or hexadecimal notation, or a symbolic name representing an IP address. Hexadecimal notation is in the form `0xaabbccdd`, where:

- `0x` is a required prefix indicating that the value is in hexadecimal format
- Each portion of the address (*aa*, *bb*, *cc*, or *dd*)
 - Is a two-digit hexadecimal value ranging from 00 to FF
 - Represents 8 bits of a 32-bit IP address

Dotted-decimal notation is in the form *a.b.c.d*, where each portion of the address (*a*, *b*, *c*, or *d*):

- Is a decimal value ranging from 0 to 255
- Is delimited by a period
- Represents 8 bits of a 32-bit IP address

Note: Do not use leading zeros to specify values in dotted-decimal notation, for example, 029.67.1.1. If you specify an integer value using one or more leading zeros, the value is interpreted as an octal value instead of a decimal value.

If you specify a symbolic name for *destination*, the name is first looked up using `gethostbyname()`. If this lookup fails, `getnetbyname()` is then used to obtain the corresponding IP address.

If *destination* can be reached directly through the `sna0` interface (with no intermediate gateways or routers):

- The address of `sna0` should be specified as *gateway*.
- The value specified as `metric` must be zero (0), indicating that a local interface is used to transmit the data.

The optional keyword `default` can be used as the *destination* parameter. Specifying `default` causes the *gateway* parameter to be interpreted as the default exit point from the local network; any IP address that does not match any other entry in the routing table is routed using the `default` entry.

gateway

Next-hop IP address to which information is routed.

Specify an IP address in dotted-decimal or hexadecimal notation. See the description under *destination* for the syntax restrictions.

The specified IP address must reside in the same subnet as the subnet defined for `sna0` using the `ISTSKRTE` utility. Otherwise, `Sockets over SNA` returns `ENETUNREACH` on the `ISTSKRTE` utility.

metric

Integer representing the number of nodes (next-hop gateways) through which information is routed before the information reaches the destination network. For example, if *metric* is set to 2, the information is routed through 2 gateways before it reaches its destination node.

Set *metric* to the value one (1) when routing information to a gateway that can be reached directly through the `sna0` interface.

Chapter 4. Configuring, Customizing, and Tuning AnyNet Sockets over SNA

This chapter describes how to:

- Configure environment variables for Sockets over SNA
- Substitute user-defined message text for fixed-message text of Sockets-over-SNA messages.

Defining Sockets-over-SNA Environment Variables: the ENVVAR Data Set

Sockets over SNA uses default values for data set names, timer intervals, and thresholds. This section describes the environment variables you can set to override these defaults. The environment variables may be defined in an EBCDIC data set. The data set is pointed to by a ENVVAR DD in the JCL used to start Sockets over SNA. The variables are formatted as follows:

environment_variable=value

where:

environment_variable

name of the environment variable for which you are setting a value

value

value you are setting for *environment_variable*.

Following is an example of an ENVVAR entry in the JCL startup procedure for Sockets over SNA. This entry is included in the JCL procedure in “Starting and Initializing Sockets over SNA” on page 35.

```
//ENVVAR DD DSN=USER1.ENVVAR,DISP=SHR
```

Notes:

1. The value you specify for each of the environment variables is case-sensitive and used as it is entered; it is not converted to uppercase.
2. All of the environment variables have default values.
3. Any environment variable can be left out of ENVVAR.
4. ENVVAR does not have to be defined at all if you want to use only default values.
5. The ENVVAR data set may contain settings for environment variable names that are not defined or supported by Sockets over SNA; these settings are ignored.
6. You do not have to define the environment variables in any particular order.

When you start Sockets over SNA, the ENVVAR data set is read for any values to use in place of default values. See “Determining Errors During Sockets-over-SNA Initialization and Operation” on page 39 for information on determining if the ENVVAR environment variables are set properly.

The following sections describe the Sockets-over-SNA environment variables you can specify in ENVVAR.

CONNECT_TIMEOUT

Default

120

Use this environment variable to define the number of seconds Sockets over SNA will wait to receive a network response to a stream socket connect () request. If an application issues a connect() for a stream socket, and Sockets over SNA does not receive a network response within CONNECT_TIMEOUT seconds, the connect() will fail with an errno value of ETIMEDOUT.

DG_CONV_IDLE_TIMEOUT

Default

5 minutes

Range

1—10080

Use this environment variable to control the time, in minutes, a datagram conversation must be unused before it is deallocated and closed. If you specify a value that is not valid, a message is issued and the default value is used.

The datagram idle timeout interval enables you to balance between using system resources to maintain an existing datagram conversation and taking longer to reestablish a new datagram conversation. For example, if you set this value low, datagram conversations end faster, but it takes longer to send the next datagram.

DG_SEGMENTATION

Default

MPTN

Alternate

LL

If a datagram is too large to be sent in one piece, the DG_SEGMENTATION environment variable determines what method is used to segment it. The MAX_SENDBUF environment variable determines how large a datagram can be before it is segmented.

Using the Default Value: MPTN

The default segmentation method, MPTN, causes a separate MPTN header to be built for each segment of the datagram. If you are communicating with AnyNet/2 Version 1.0 Sockets over SNA or with AnyNet/2 Sockets over SNA Gateway Version 1.0, you **must** use MPTN segmentation.

Using the Alternate Value: LL

Specifying LL causes a single MPTN header to be sent for the datagram. APPC's native segmentation method, LL, is used to split the datagram into smaller segments. Depending on characteristics of the application program and your network, LL segmentation may be more efficient.

HOSTNAME

Use this environment variable to specify the host name of the local machine. This name is returned by the gethostname() call. If you do not use this environment variable to specify a local hostname, gethostname() returns the string "localhost".

LEARNMAP = NOYES

Default

YES

When LEARNMAP is YES, a z/OS host will deduce the IP-LU mapping for all inbound connections and datagrams, and update the IP-LU table if necessary. Therefore, if a z/OS host is a server, the IP-LU mapping table entries for all clients that initiate connections to the host need not be configured explicitly into the IP-LU table using ISTSKMAP. Of course, each client needs to configure an IP-LU mapping for the z/OS host server. Furthermore, an IP-LU mapping must be configured on the z/OS host for any client the z/OS host initiates traffic to. The host cannot deduce the mapping if it has no prior inbound traffic from the client.

You must configure IP-LU mappings for gateway nodes. The host will not change an IP-LU mapping for a gateway node.

When the z/OS host updates the IP-LU table with a new mapping, the IP-LU table becomes larger. The z/OS host will not add an entry to the table if it is already there, or if the mapping can be generated algorithmically from one of the templates in the IP-LU mapping table. The larger the table, the slower the IP-LU look-ups, so you may find it more efficient to configure your table with IP-LU map templates that generate most or all the IP-LU mappings for the clients you expect your host to serve.

When LEARNMAP is NO, an IP-LU mapping must be configured for all nodes the z/OS host communicates with.

Note: When LEARNMAP is NO, only those nodes configured into the z/OS host can communicate with that host. When LEARNMAP is YES, any MPTN access node can communicate with the host.

MAX_CONCURRENT_SOCKET_CALLS

Default

100

Minimum allowed value

10

Maximum allowed value

1500

This environment variable controls the number of socket API calls Sockets over SNA can handle at once.

This is not the same as the maximum number of socket calls for z/OS UNIX, because z/OS UNIX will route some calls to other transport providers. Only socket API calls handled by Sockets over SNA count towards this total.

If an application attempts more than MAX_CONCURRENT_SOCKET_CALLS socket calls at once, the excess calls will fail with an errno value of EAGAIN.

Each concurrent socket call requires Sockets over SNA to allocate storage. Some socket calls require Sockets over SNA to allocate more storage than others, and increase storage allocated to Sockets over SNA when MAX_CONCURRENT_SOCKET_CALLS are being processed. The higher you set

MAX_CONCURRENT_SOCKET_CALLS, the more storage you will need to provide to your Sockets over SNA job to allow it to attain MAX_CONCURRENT_SOCKET_CALLS calls.

MAX_DG_CONVS

Default

10

Range

1—32768

This environment variable controls the number of outgoing datagram conversations Sockets over SNA allows to be open simultaneously. If you specify a value that is not valid, a message is issued and the default value is used.

Sockets over SNA checks every 5 seconds to ensure that the number of datagram conversations does not exceed the specified value. If the number of datagram conversations exceeds the specified value, the conversations are deallocated. When setting this environment variable, consider the trade-off between using system resources to maintain idle conversations and using system resources to reallocate conversations.

MAX_SENDBUF

Default

8300 bytes

Range

1000—32767

Specifies the maximum number of bytes, including MPTN headers generated by Sockets over SNA, that Sockets over SNA sends at a time.

If the size of the data plus the MPTN header is larger than the value specified as MAX_SENDBUF,

- For stream connections, Sockets over SNA splits the data and uses multiple sends.
- For datagram connections, Sockets over SNA segments the data using the value specified as the DG_SEGMENTATION parameter.

Using the Default Value: 8300

If you are communicating with AnyNet/2 Version 1.0 Sockets over SNA or with AnyNet/2 Sockets over SNA Gateway Version 1.0, you **must** specify a value between 1000 and 8300.

Using an Alternate Value

If a Sockets-over-SNA application program sends data in portions that exceed the default (or specified) value, increasing this environment variable to a value that eliminates the need to segment the data could improve system throughput. It is unlikely that you would ever need to lower the buffer size to less than 8300.

OE_INADDRANY_COUNT

Default

None

Range

1—4000

This environment variable specifies the number of ports that will be reserved for z/OS UNIX INADDR_ANY binds starting with the port number specified with the OE_INADDRANY_PORT environment variable. Use this environment variable only if you are using z/OS UNIX common INET. The values should match the value of INADDRYANY_COUNT in the BPXPRMxx parmlib member.

If you code OE_INADDRANY_COUNT, you must also code OE_INADDRANY_PORT. Furthermore, if another product is also a transport provider to z/OS UNIX, that product must reserve the same ports. Refer to each product's documentation for information on reserving ports for INADDR_ANY binds.

OE_INADDRANY_PORT

Default

None

Range

1024—65534

This environment variable specifies the starting port number for the range of port numbers that will be reserved by the system for exclusive use on INADDR_ANY bind() requests received from z/OS UNIX. Use this environment variable only if you are using common INET. The values should match the value of INADDRYANY_PORT in the BPXPRMxx parmlib member.

If you code OE_INADDRANY_PORT, you must also code OE_INADDRANY_COUNT. If another product is also a transport provider to z/OS UNIX, that product must reserve the same ports. Refer to each product's documentation for information on reserving ports for INADDR_ANY binds.

SXMODE_DEFAULT

Default

SNACKETS

Use this environment variable to override the default mode name (SNACKETS) used by Sockets over SNA. The mode name you specify becomes the default for all ports established for Sockets-over-SNA communication.

See “Defining a MODEENT Macro Entry for Sockets over SNA” on page 16 for more information on specifying a mode entry for all Sockets-over-SNA application programs.

SXMODEn

Use this environment variable to specify a mode for stream connections over an individual port. Note that ports using datagram connections cannot use a mode value other than the default mode in effect for all ports. The name you specify overrides the default mode name (SNACKETS) applying to all ports. The last letter of this environment variable, *n*, is a positional character representing the port value that you append to the name. Following are examples of how you would use SXMODE*n* to specify mode SNACK23 for port 23 and mode SNACKSPC for port 1022. Port values in the following examples are in decimal format with no leading zeros.

SXMODE23=SNACK23
SXMODE1022=SNACKSPC

See “Defining a MODEENT Macro Entry for Sockets over SNA” on page 16 for more information on specifying a mode entry for a specific port.

TRACE_OPTION

Default

off all

This environment variable traces options you can specify during initialization. Besides the default, the other possible values are any parameters passed to IOSTKTRC other than reset, such as “on all” or “on sic”.

Routing ENVVAR Messages

ENVVAR messages generated during startup are routed to the standard output file. If you want to route ENVVAR-setting messages to a file other than the standard output file, define a file with a DD name of LOGMSGDS in the job (or procedure) used to start Sockets over SNA. Following is an example of a JCL entry to reroute messages:

```
//LOGMSGDS DD SYSOUT=*
```

Customizing Message Text for Sockets-over-SNA Messages

You can alter the fixed message text of Sockets-over-SNA messages according to requirements of your installation. Using the MVS message service is the best way to customize messages. However, if you prefer to change Sockets-over-SNA message text yourself, edit the source data set, IOSTKDMM, in SYS1.SAMPLIB. IOSTKDMM contains the message text and the JCL for compiling and linking. To make your changes take effect, compile the edited source data set and link edit to the new load module. Be aware that every time you change the message text, you must also recompile and relink IOSTKDMM. In addition, make sure you back up the source data set and load module before editing.

Refer to the *z/OS Language Environment Programming Guide* for more information on compiling and linking C programs.

Defining the Sockets-over-SNA Transport Driver to z/OS UNIX System Services

To define Sockets-over-SNA transport driver to z/OS UNIX:

- The initialization routine entry point ISTOEPIT must be defined and reside in the STEPLIB of the OMVS cataloged procedure or the system link list.
- The Sockets-over-SNA file system must be defined to z/OS UNIX via the parmlib member BPXPRMxx, which is specified when the z/OS UNIX address space is started.
- If you are not using the common INET PFS, the transport driver is defined using the FILESYSTYPE definition statement, and the ENTRYPOINT parameter on the Sockets-over-SNA file system statement specifies ISTOEPIT as the parameter. An example follows:

```
FILESYSTYPE TYPE(APPNSNA)  
ENTRYPOINT(ISTOEPIT)
```

The NETWORK definition statement that corresponds to the Sockets-over-SNA FILESYSTYPE statement should specify a DOMAINNAME of AF_INET and a DOMAINNUMBER of 2. The following NETWORK definition statement corresponds to the previous FILESYSTYPE definition statement:

```
NETWORK TYPE(APPNSNA)
        DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(32)
```

- If you are using the common INET PFS, the NAME(_) on SUBFILESYSTYPE must match the proc name of the job used to start Sockets over SNA.

An example follows:

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTICINT)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(32)
        TYPE(CINET)
        INADDRANYPORT(2000)
        INADDRANYCOUNT(100)
SUBFILESYSTYPE NAME(SNACKOE)
        TYPE(CINET)
        ENTRYPOINT(ISTOEPIT)
        DEFAULT
SUBFILESYSTYPE NAME(TCPIP)
        TYPE(CINET)
        ENTRYPOINT(BPXTIINT)
```

- The TYPE parameters on the FILESYSTYPE, SUBFILESYSTYPE and NETWORK definition statements should be the same. The TYPE parameter correlates the definition statements.
- For security authorization, Sockets over SNA must be authorized to your security product before you can use the z/OS UNIX services. For example, if you use the IBM security product RACF, Sockets over SNA must be RACF-authorized to use the z/OS UNIX callable services.

Restrictions and Constraints

The following restrictions and constraints apply to defining the Socket-over-SNA transport driver to z/OS UNIX for MVS:

- Without common INET, only a single FILESYSTYPE definition statement with ENTRYPOINT(ISTOEPIT) can be specified. If additional FILESYSTYPE definition statements with ENTRYPOINT(ISTOEPIT) are specified, the subsequent physical file systems will abend with abend code X'CC5' and reason code X'xxxx0847'.
- Use the following Sockets-over-SNA environment variables in the ENVVAR data set to specify which ports are reserved for INADDRANY binds from z/OS UNIX:
 - OE_INADDRANY_PORT
 - OE_INADDRANY_COUNT.

The common INET transport driver ensures that certain ports are available on all INET TPs for INADDRANY_PORT and INADDRANY_COUNT. Within Sockets over SNA, the ports will be reserved using these environment variables.

There are no default values for these environment variables. These values must be coordinated with other z/OS UNIX transport providers, such as TCP/IP. For more information about these environment variables, see “OE_INADDRANY_PORT” on page 31 and “OE_INADDRANY_COUNT” on page 30.

Chapter 5. Starting, Initializing, and Stopping AnyNet Sockets over SNA

This chapter describes the following tasks associated with starting, initializing, and using Sockets over SNA.

- Starting Sockets over SNA
- Initializing Sockets over SNA and running an application
- Stopping Sockets over SNA

Before you attempt to start and use Sockets over SNA, make sure you have finished the planning, installation, and configuration tasks described in “Chapter 2. Planning for AnyNet Sockets over SNA” on page 7, “Chapter 3. Defining AnyNet Sockets over SNA” on page 15, and “Chapter 4. Configuring, Customizing, and Tuning AnyNet Sockets over SNA” on page 27.

Order of Initialization

Following is the order for starting Sockets over SNA, a Sockets-over-SNA application program, and the supporting system components:

1. IPL the operating system. This step starts z/OS UNIX and the Sockets-over-SNA physical file system.
2. Start VTAM.
3. Activate the APPL major node containing the Sockets-over-SNA definition.
4. Start Sockets over SNA.
5. Run the ISTSKMAP utility to initialize the IP-LU mapping table. This step has to precede using ISTSKIFC to initialize the Sockets-over-SNA interface.
6. Run the ISTSKIFC utility to initialize the sna0 interface.
7. Run the ISTSKIFC utility to initialize the 1o interface.
8. Start the socket application.
9. Monitor your operator's console and standard output device to ensure that Sockets over SNA has initialized successfully. If all the software components start up with no failures, resume normal operation procedures.

Starting and Initializing Sockets over SNA

This section provides sample JCL code for starting Sockets over SNA and the ISTSKMAP, ISTSKIFC, and ISTSKTRC utilities. You can start Sockets over SNA, ISTSKMAP, ISTSKIFC, and ISTSKTRC as procedures or as jobs.

It is recommended that you specify a region size of 8 MB (REGION=8M) or more when you initialize Sockets over SNA.

During operation, data is queued in the Sockets-over-SNA address space. If your installation experiences long response delays or transfers large volumes of data, you might need to increase the region size of the Sockets-over-SNA address space to improve performance.

Following is sample JCL code for starting Sockets over SNA as a procedure:

```
//SNCK      PROC
//DAEMON    EXEC PGM=ISTSKDMN,REGION=0M,TIME=1440,
//          PARM='ENVAR(_CEE_ENVFILE=DD:ENVVAR)'
//STEPLIB   DD DSN=SYS1.VTAMLIB,DISP=SHR
```

```
//          DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD DUMMY
//ENVVAR   DD DSN=ANYNET.CFG(ENVVAR),DISP=SHR
```

Notes:

1. Specify at least an 8 megabyte address space for Sockets over SNA (REGION=8M).
2. The Sockets-over-SNA load module resides in SYS1.VTAMLIB.
3. ISTSKDMN is the main Sockets-over-SNA load module.
4. SCEERUN is the Language Environment** runtime library.
5. CEEDUMP is the data set used to capture information related to situations that should not occur.
6. ANYNET.CFG(ENVVAR) is the ENVVAR data set.
7. SYSPRINT is the default trace data set.

After you start Sockets over SNA, use ISTSKMAP and ISTSKIFC to initialize the IP-LU mapping table and to assign an address to the sna0 interface. You must run ISTSKMAP and ISTSKIFC before you start a Sockets-over-SNA application program (see “Using the ISTSKMAP Utility to Establish, Update, and Query the Mapping Table” on page 18 and “Using the ISTSKIFC Utility to Define an SNA Network Interface for Sockets over SNA” on page 22). Following is sample code that starts the ISTSKMAP, ISTSKIFC, ISTSKNST, ISTSKRTE, and ISTSKTRC utilities procedure:

```
//SETUP      PROC
//*-----
//* THIS JOB STEP TURNS ON ALL SNACKETS TRACES.
//*-----
//TRACEON    EXEC PGM=ISTSKTRC,PARM=('ON ALL'),REGION=1M
//STEPLIB    DD DSN=SYS1.VTAMLIB,DISP=SHR
//           DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//*-----
//* THIS JOB STEP MAPS IP ADDRESSES TO NETID AND LUNAMES
//*-----
//MAP        EXEC PGM=ISTSKMAP,REGION=1M,
//           PARM=('ADD 128.109.0.0 255.255.0.0 NETA SX')
//STEPLIB    DD DSN=SYS1.VTAMLIB,DISP=SHR
//           DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//*-----
//* THIS JOB STEP CONFIGURES THE LOOPBACK INTERFACE
//*-----
//LOCONFIG   EXEC PGM=ISTSKIFC,PARM=('LO 127.0.0.1'),REGION=1M
//STEPLIB    DD DSN=SYS1.VTAMLIB,DISP=SHR
//           DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//*-----
//* THIS JOB STEP CONFIGURES THE SNA0 INTERFACE
//*-----
//ISTSKIFC   EXEC PGM=ISTSKIFC,REGION=1M,
//           PARM=('SNA0 128.109.255.202')
//STEPLIB    DD DSN=SYS1.VTAMLIB,DISP=SHR
//           DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//*-----
//* THIS JOB STEP ADDS AN ENTRY TO THE ROUTING TABLE
//* FOR HOST 129.34.10.1 TO USE GATEWAY 128.109.255.204
//*-----
//ISTSKRTE   EXEC PGM=ISTSKRTE,REGION=1M,
//           PARM=('ADD HOST 129.34.10.1 128.109.255.204 1')
//STEPLIB    DD DSN=SYS1.VTAMLIB,DISP=SHR
```

```
//          DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
//*-----
/* THIS JOB STEP ADDS AN ENTRY TO THE ROUTING TABLE
/* FOR NETWORK 129.34 (CLASS B) TO USE GATEWAY 128.109.255.204
/*-----
//ISTSKRTE EXEC PGM=ISTSKRTE,REGION=1M,
//          PARM=('ADD NET 129.34 128.109.255.204 1')
//STEPLIB DD DSN=SYS1.VTAMLIB,DISP=SHR
//          DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
//*-----
/* THIS JOB STEP DISPLAYS THE ROUTING TABLE, SOCKET TABLE AND
/* INTERNAL STATISTICS FOR SOCKETS OVER SNA
/*-----
/ISTSKNST EXEC PGM=ISTSKNST,REGION=1M,
/          PARM=(' -R -S -D')
/STEPLIB DD DSN=SYS1.VTAMLIB,DISP=SHR
/          DD DSN=LE370.V1R8M0.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Notes:

1. The ISTSKMAP, ISTSKIFC, ISTSKNST, and ISTSKTRC load modules reside in SYS1.VTAMLIB.
2. The REGION parameter is optional. If you specify REGION, specify a region of at least 1M.
3. The procedure must be RACF authorized. Refer to “Running Utilities Using JCL” for an example of how to supply a RACF authorized userid and password when running utilities from JCL.

Refer to the *z/OS MVS JCL User's Guide* for more information on running programs using JCL. Refer to the *z/OS Language Environment Programming Guide* for more information on running programs that use the LE/370 runtime library.

Running Utilities Using JCL

You must supply a RACF authorized userid and password when running the utilities using JCL, as shown in the second line of the following example.

To run Sockets-over-SNA utilities using JCL, submit the following:

```
//TROFFTO JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),REGION=0M,
//          USER=USER1,PASSWORD=password
//JOB LIB DD DSN=SYS1.VTAMLIB,DISP=SHR
//          DD DSN=SYS1.SCEERUN,DISP=SHR
//*
//TRACEOFF EXEC PGM=ISTSKTRC,PARM=('off timo')
//SYSPRINT DD SYSOUT=*
//SNLOUT DD SYSOUT=*
//*
```

Running Utilities from the z/OS UNIX Shell

To run Sockets-over-SNA utilities from the z/OS UNIX shell, execute the following from the z/OS UNIX shell command line:

```
export STEPLIB="SYS1.VTAMLIB"
```

This adds VTAMLIB to the z/OS UNIX shell search string. Next, create a file with the sticky bit attribute that has the same name as the utility that you want to execute. For example, to execute ISTSKNST, create a file called ISTSKNST. Use the CHMOD command to turn the sticky bit on and mark the file executable:

```
echo    this is a pointer to load module ISTSKNST >ISTSKNST  
chmod +tx ISTSKNST
```

Refer to the *z/OS UNIX System Services User's Guide* for an explanation of sticky bit and the chmod and echo commands.

The following command will enter a load module name and arguments on a z/OS UNIX command line:

```
ISTSKNST -d
```

Stopping Sockets over SNA

To stop the execution of Sockets over SNA, use the MVS STOP command. Refer to *z/OS MVS System Commands* for more information on the STOP command.

Chapter 6. Diagnosing Problems for AnyNet Sockets over SNA

This chapter describes:

- How you can use console and system log messages to help determine definition and configuration errors
- How to use the ISTSKNST utility to display Sockets-over-SNA routing tables
- Traces you can run to help diagnose problems that might arise when you install, configure, or use Sockets over SNA
- How to respond to abends caused by Sockets over SNA

The status of Sockets over SNA, associated subsystems, and associated application programs is indicated using the following methods:

Console Messages

Sockets over SNA issues ISU-prefixed console messages for error situations where operator intervention might be needed and for status milestones that report the progress of initialization and shutdown. ISU-prefixed messages are described in *z/OS Communications Server: SNA Messages*.

Log Messages

Sockets over SNA issues log messages to inform you when certain values have been set. For example, when Sockets over SNA starts, values used for ENVVAR parameters are reported. Informational log messages are directed to the standard output device.

Errno Values

When an application program begins communication with Sockets over SNA, errors that occur are usually mapped to the errno global variable.

Determining Errors During Sockets-over-SNA Initialization and Operation

Message ISU1501I is issued when Sockets over SNA starts successfully. The ISTSKMAP utility is then used to initialize the IP-LU mapping table, causing the arguments you specify to echo to ISTSKMAP's standard output. If Sockets over SNA detects a mapping error during normal operation, either of the following two message groups are generated. Note that these message groups are generated by Sockets over SNA, not by ISTSKMAP.

- If Sockets over SNA is unable to map an IP address to an LU name, message ISU1517I is generated. If ISU1517I is preceded by ISU1519I, the IP-LU mapping table does not contain an entry that matches the IP address contained in the message.
- If Sockets over SNA detects a mapping inconsistency between the LU name generated by the IP-LU mapping algorithm and the LU name used by a partner, messages ISU1534I and ISU1535I are generated.

The next step, using ISTSKIFC to initialize the Sockets-over-SNA interface, does not echo any arguments you specify. If Sockets over SNA detects an error attempting to configure the sna0 interface, the name of the macro function that caused the failure is included in the message group generated for message ISU1519I. Refer to message ISU1519I in *z/OS Communications Server: SNA Messages* for complete information on message ISU1519I.

If an application program issues an API call that results in an error, the API call indicates failure and the global `errno` variable is set to indicate the cause of the failure. A typical failure scenario is `EAGAIN`:

EAGAIN

You tried to start your application before Sockets over SNA completed its initialization.

EIO Sockets over SNA was stopped while processing the socket call.

For more error node values and meanings, refer to the *z/OS UNIX System Services Programming: Assembler Callable Services Reference* or the *z/OS C/C++ Run-Time Library Reference*.

Determining Whether Environment Variables Are Set Successfully

During initialization, Sockets over SNA writes all environment variable values to the standard output file (`SYSPRINT`), or to `DD:LOGMSG`, if defined. To ensure that the environment variable values you want are being used, examine the `SYSPRINT` output during initialization. Following are examples of log messages that go into standard output indicating environment variable values for Sockets-over-SNA initialization:

```
Sockets-over-SNA: Log message table initialized
Sockets-over-SNA: Message table initialized
Sockets-over-SNA: ENVVAR settings in effect:
Sockets-over-SNA: TRACE_OPTION is on all
Sockets-over-SNA: MAX_CONCURRENT_SOCKET_CALLS is 12
Sockets-over-SNA: HOSTNAME is SNACKETS.RALEIGH.IBM.COM
Sockets-over-SNA: DG_SEGMENTATION is MPTN
Sockets-over-SNA: SXMODE_DEFAULT is SNACKETS
Sockets-over-SNA: MAX_DG_CONVS is 10
Sockets-over-SNA: DG_CONV_IDLE_TIMEOUT is 5
Sockets-over-SNA: MAX_SENDBUF is 8300
Sockets-over-SNA: CONNECT_TIMEOUT is 120
Sockets-over-SNA: OE_INADDRANY_PORT is 2000
Sockets-over-SNA: OE_INADDRANY_COUNT is 100
Sockets-over-SNA: LEARNMAP is YES
```

If during initialization, Sockets over SNA detects an environment variable value that is not valid, the default value for that environment variable is used and Sockets over SNA continues the initialization process. There are two exceptions: `OE_INADDRANY_PORT` and `OE_INADDRANY_COUNT`, which are ignored because they do not have a default value.

Messages `ISU1512I`, `ISU1568I`, `ISU1569I`, and `ISU1570I` during initialization indicate an invalid environment variable.

Refer to *z/OS Communications Server: SNA Messages* for complete information on all Sockets-over-SNA messages.

Using the `ISTSKNST` Utility to Display Sockets-over-SNA Routing Tables

This section describes how to use the `ISTSKNST` utility to display entries in the local Sockets-over-SNA routing table. This version of the `ISTSKNST` utility is provided to help you verify routes that are established through one or more Sockets Gateways. (Refer to *IBM Communications Server for OS/2 Warp: Guide to AnyNet Sockets over SNA* for information on using Sockets Gateways.)

The ISTSKNST utility can help you determine:

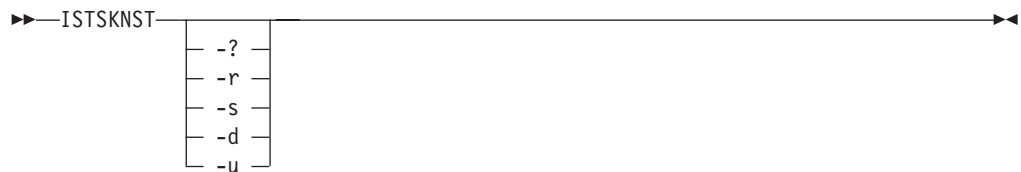
- Whether a route exists to a specific destination node or destination network
- Which gateway is used for a given route
- Which gateway is defined for the default route
- If a specific route has been used.
- An appropriate value for MAX_CONCURRENT_SOCKET_CALLS

The ISTSKNST utility can be run either interactively (for example, using TSO), or using batch (for example, JCL). To run ISTSKNST interactively, your logon procedure must be set up to run an authorized LE/370[®] application programs. Refer to “Domain Name Server” on page 13 for information on defining a domain name server. Refer to your *z/OS Language Environment Programming Guide* for information on setting up your logon procedure to run LE/370 application programs.

Following is an example of how to run ISTSKNST in TSO with the -? argument for a quick view of the usage syntax.

```
call 'sys1.vtamlib(istsknst)' '-?'
```

The syntax for the ISTSKNST utility follows:



Following is a description of each operand:

(-? or no operand)

If you specify the ISTSKNST utility with a question mark, unrecognizable operand, or no operand, ISTSKNST displays the following usage syntax:

Usage: ISTSKNST [-r] [-s] [-d] [-u]

Where:

- | | |
|----|---|
| -r | displays the entries in the routing table |
| -s | displays the socket table |
| -d | displays statistics |
| -u | updates the displays every two seconds |

- r** displays the following information for all route entries in the Sockets-over-SNA routing table.

destination

IP address of the node or network that can be reached through the IP address displayed as gateway

gateway

IP address of the node through which the *destination* node or *destination* network can be reached

refcnt

Current number of active uses of the route. Connection-oriented protocols normally hold on to a single route for the duration of a connection; connectionless protocols obtain a route while sending data.

metric

The number of hops to destination. If metric is zero, it is a local interface. If it is greater than zero, it is non-local.

use

Number of packets that have been sent using this route

flags

U indicates this route is connected, active, and available for use.

D indicates this route was created dynamically.

M indicates this route has been modified.

G indicates this route is a gateway.

H indicates this is a host.

intrf

indicates the network interface that is used to reach the specified gateway.

On z/OS, if Sockets over SNA is configured correctly, these fields are set to sna0 or lo.

Following is an example of the output produced when you run ISTSKNST and specify the -r:

destination	gateway	refcnt	metric	flags	intrf
129.34.10.1	128.109.255.204	0	1	UGH	sna0
127.0.0.1	127.0.0.1	0	0	UH	lo
129.34.10.0	128.109.255.205	0	1	U	sna0
128.109.0.0	128.109.255.202	0	0	U	sna0

-s displays the socket table.

Following is an example of the output produced when you run ISTSKNST and specify the -s:

SOCK	TYPE	FOREIGN PORT	LOCAL PORT	FOREIGN HOST	STATE
4	STREAM	0	0	0.0.0.0	

Socket descriptors are those assigned by Sockets over SNA. z/OS UNIX System Services gives the application a different descriptor.

-d displays the statistics.

Following is an example of the output produced when you run ISTSKNST and specify the -d:

2 out of a maximum 10 environments are allocated.

This output shows you have configured MAX_CONCURRENT_SOCKET_CALLS to be 10; however, Sockets over SNA has only executed 2 calls concurrently since Sockets over SNA was started.

If the output shows all environments have been allocated, consider increasing MAX_CONCURRENT_SOCKET_CALLS.

Using the OPING Utility

Earlier releases of Sockets over SNA provided the ISTSKPNG, or ping, utility. In this release, OPING is provided by z/OS Communications Server.

Using the ISTSKTRC Utility to Trace Sockets over SNA

The ISTSKTRC utility can be run either interactively (for example, using TSO), or using batch (for example, JCL). To run ISTSKTRC interactively, your logon procedure must be set up to run an authorized LE/370[®] application programs. Refer to “Domain Name Server” on page 13 for information on defining a domain name server. Refer to your *z/OS Language Environment Programming Guide* for information on setting up your logon procedure to run LE/370 application programs.

Following is an example of how to run ISTSKTRC in TSO with all trace options on.

```
call 'sys1.vtamlib(istsktrc)' 'on all'
```

Syntax for Running ISTSKTRC

The ISTSKTRC utility is used to turn tracing on and off for the Sockets-over-SNA address space. You can select the categories of information you want to trace by specifying one or more trace options. Your IBM support representative can help you determine what trace categories to trace.

The syntax for the ISTSKTRC utility follows:



ISTSKTRC

required utility used to activate ISTSKTRC

on | off

required argument that turns tracing on or off. If you specify one or more trace options, turns tracing on or off for the specified options.

reset

optional argument that turns off all tracing and closes the trace output file. Do not specify a trace option when you use reset.

Reset is the only way to empty the trace buffer. If you restart tracing, previous trace information may be overwritten.

trace_options

category of information you want to trace. *trace_options* is optional. You can specify one or more of the following trace options.

Notes:

1. If you do not specify at least one trace option, all categories are traced.
2. Separate each option with a space.
3. Options are not case sensitive.

Trace Option

	Description
TCP	streams
UDP	datagrams
TIMO	timeout routines
MISC	miscellaneous
GEN	gen_usrreq function
INP	input
OUT	output
SIC	socket interface calls
DATI	data in

DATO	data out
ASY	asynchronous events
APPC	APPC calls
SLP	sleep and wakeup calls
SIA	socket interface arguments
MEM	memory usage
MSG	console messages
IP	IP
DIE	termination events
ALL	traces all categories
OE	z/OS UNIX events

–q active trace options are listed when ISTSKMAP completes execution unless –q is specified. This argument is optional.

Routing Output from the ISTSKTRC Utility

Unless you use one of the methods shown below, Sockets over SNA sends trace data to the SYSPRINT DD of the job or PROC used to start Sockets over SNA.

If an error is encountered writing trace data to the standard output file, for example, SYSPRINT fills with data, Sockets over SNA continues running but stops writing trace data to SYSPRINT. If you want to route ISTSKTRC output to a file other than the standard output file, use either of the following methods:

- Define a file with a DD name of TRACE in the job (or procedure) used to start Sockets over SNA. If an error is encountered while writing to TRACE (for example, if the TRACE file fills with data), Sockets over SNA stops writing trace data to the file and reverts to using SYSPRINT. Following is an example of a JCL entry to trace to a small data set, which will eventually fill up:

```
//TRACE DD DSN=USER.TRACE
```

- To trace to spool instead of to DASD, replace the TRACE DD card with the following card:

```
//SYSPRINT DD SYSOUT=*
```

Note: Tracing to spool can create a large amount of printed output and, if run for any length of time, can fill up the spool. Another method of tracing to spool is to encode the TRACE DD card to direct output to spool:

```
//TRACE DD SYSOUT=*
```

- Define one or more files with a DD name of TRACE n , where n is an integer from 1 to 999. Following is an example of JCL entries that route trace messages to 3 TRACE n entries:

```
//TRACE1 DD DSN='USERX.TRACE1',DISP=SHR
//TRACE2 DD DSN='USERX.TRACE2',DISP=SHR
//TRACE3 DD DSN='USERX.TRACE3',DISP=SHR
```

If an error is encountered while writing to any of the TRACE n files, Sockets over SNA stops writing to the file and goes to the next TRACE n file in the sequence. Writing continues until an error occurs attempting to write to file TRACE $n+1$ (for example, the last specified TRACE n file fills with trace data). After TRACE n fills up, the process starts over and trace entries are written to TRACE1.

If an ISTSKTRC reset is issued during tracing, the current TRACE n data set is closed. If tracing is restarted, Sockets over SNA starts writing trace data to TRACE1.

Using TRACE n data sets enables you to collect the most current trace information and limit the amount of past trace information that is stored.

If your trace card and/or TRACE DD card prints to a data set (DASD) then the data set name must be an HFS file name or a z/OS data set of fixed format with logical record length of at least 150 (LRECL must be greater than or equal to 150 and RECF must be equal to F).

See “Starting and Initializing Sockets over SNA” on page 35 for sample JCL that starts the ISTSKTRC utility and sends the output to either a standard output device (SYSPRINT) or a user-specified data set.

If you use ISTSKTRC reset to turn off tracing and close the file, the file is reopened and the entries are overwritten when tracing is resumed. If you want to save the contents of the trace file before you resume tracing, copy the file after you issue ISTSKTRC reset.

Responding to Sockets-over-SNA Abends

Sockets over SNA issues abends to z/OS and requests dumps if an unrecoverable error occurs.

If Sockets over SNA abends without creating a dump, you can obtain abend-related information by dumping the Sockets-over-SNA address space just as you would dump any other address space under z/OS.

Sockets over SNA detects some abend situations and attempts to bring itself down. See *z/OS Communications Server: SNA Messages* for a listing of the abends that can be generated by failures related to Sockets over SNA.

Service Information

If you cannot determine the cause of the problem from the output provided, take the following actions:

- If you have access to IBMLink™, search for known problems in this area. If no applicable matches are found, report the problem to IBM by using the Electronic Technical Report (ETR) option on IBMLink.
- If you do not have access to IBMLink, report the problem to the IBM Support Center.

Appendix A. Syntax Conventions

This appendix describes the syntax conventions that are used in this book.

To help you locate and identify information easily, this book uses the following visual cues and standard text formats:

Font Type

Used for

monospace

Commands, command options, examples, messages, text that you type or enter

CAPS File names and file paths

italics User-supplied variables, message variables, titles of books

bold Socket call names, return codes

“quotes”

Titles in a menu

How to Read a Syntax Diagram

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and Punctuation

The following symbols are used in syntax diagrams:

- ▶▶ Marks the beginning of the command syntax.
- ▶ Indicates that the command syntax is continued.
- | Marks the beginning and end of a fragment or part of the command syntax.
- ◀◀ Marks the end of the command syntax.

You must include all punctuation such as colons, semicolons, commas, quotation marks, and minus signs that are shown in the syntax diagram.

Parameters

The following types of parameters are used in syntax diagrams.

Required

Required parameters are displayed on the main path.

Optional

Optional parameters are displayed below the main path.

Default

Default parameters are displayed above the main path.

Parameters are classified as keywords or variables. Keywords are displayed in uppercase letters and can be entered in uppercase or lowercase. For example, a command name is a keyword.

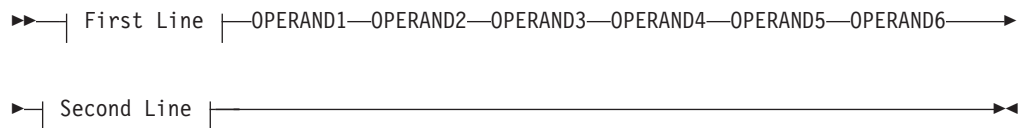
Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

Syntax Examples

In the following example, the `USER` command is a keyword. The required variable parameter is `user_id`, and the optional variable parameter is `password`. Replace the variable parameters with your own values.



Longer than one line: If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.

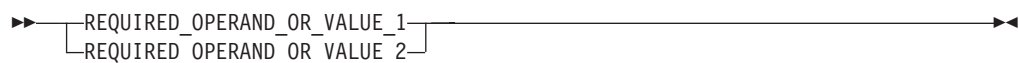


Required operands: Required operands and values appear on the main path line.



You must code required operands and values.

Choose one required item from a stack: If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically in alphanumeric order.



Optional values: Optional operands and values appear below the main path line.



You can choose not to code optional operands and values.

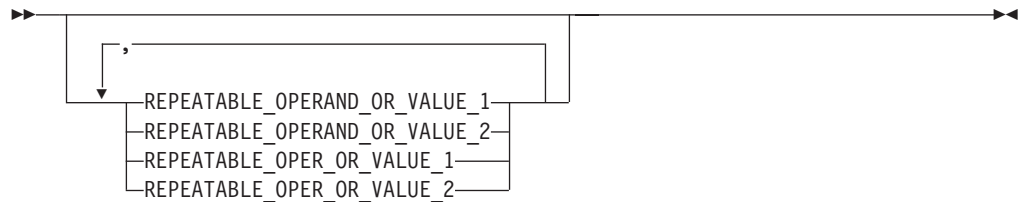
Choose one optional operand from a stack: If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically in alphanumeric order below the main path line.



Repeating an operand: An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The command means that each operand or value must be separated from the next by a comma.



Selecting more than one operand: An arrow returning to the left above a group of operands or values means more than one can be selected, or a single one can be repeated.



If an operand or value can be abbreviated, the abbreviation is described in the text associated with the syntax diagram.

Case Sensitivity: VTAM commands are not case sensitive. You can code them in uppercase or lowercase.

Nonalphanumeric characters: If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code `OPERAND=(001,0.001)`.



Blank spaces in syntax diagrams: If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code `OPERAND=(001 FIXED)`.



Default operands: Default operands and values appear above the main path line. VTAM uses the default if you omit the operand entirely.



Variables: A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

▶▶—*variable*—▶▶

Syntax fragments: Some diagrams contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

▶▶| Reference to Syntax Fragment |▶▶

Syntax Fragment:

|—1ST_OPERAND,2ND_OPERAND,3RD_OPERAND—|

References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

(1)
▶▶—OPERAND—▶▶

Notes:

- 1 An example of a syntax note.

Appendix B. Information Apars

This appendix lists information apars for IP and SNA books.

Notes:

1. Information apars contain updates to previous editions of the manuals listed below. Books updated for V1R2 are complete except for the updates contained in the information apars that may be issued after V1R2 books went to press.
2. Information apars are predefined for z/OS V1R2 Communications Server and may not contain updates.

IP Information Apars

Table 4 lists information apars for IP books.

Table 4. IP Information Apars

Title	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8	CS for OS/990 2.7	CS for OS/390 2.6	CS for OS/390 2.5
IP API Guide	ii12861	ii12371	ii11635	ii11558	ii11405	ii11144
IP CICS Sockets Guide	ii12862		ii11626	ii11559	ii11406	ii11145
IP Configuration			ii11620 ii12068 ii12353 ii12649	ii11555 ii11637 ii11995 ii12325	ii11402 ii11619 ii12066 ii12455	ii11159 ii11979 ii12315
IP Configuration Guide	ii12498	ii12362 ii12493				
IP Configuration Reference	ii12499	ii12363 ii12494 ii12712				
IP Diagnosis	ii12503	ii12366 ii12495	ii11628	ii11565	ii11411	ii11160 ii11414
IP Messages Volume 1	ii12857	ii12367	ii11630	ii11562	ii11408	ii11636
IP Messages Volume 2	ii12858	ii12368	ii11631	ii11563	ii11409	ii11281
IP Messages Volume 3	ii12859	ii12369	ii11632 ii12883	ii11564 ii12884	ii11410 ii12885	ii11158
IP Messages Volume 4	ii12860					
IP Migration	ii12497	ii12361	ii11618	ii11554	ii11401	ii11204
IP Network Print Facility	ii12864		ii11627	ii11561	ii11407	ii11150
IP Programmer's Reference	ii12505		ii11634	ii11557	ii11404	ii12496

Table 4. IP Information Apars (continued)

Title	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8	CS for OS/990 2.7	CS for OS/390 2.6	CS for OS/390 2.5
IP and SNA Codes	ii12504	ii12370	ii11917	Added TCP/IP codes to VTAM codes V2R6 ii11611	ii11361	ii11146 ii11097
IP User's Guide		ii12365	ii11625	ii11556	ii11403	ii11143
IP User's Guide and Commands	ii12501					
IP System Admin Guide	ii12502					
Quick Reference	ii12500	ii12364				

SNA Information Apars

Table 5 lists information apars for SNA books.

Table 5. SNA Information Apars

Title	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5
Anynet SNA over TCP/IP			ii11922	ii11633	ii11624	ii11623
Anynet Sockets over SNA			ii11921	ii11622	ii11519	ii11518
CSM Guide						
IP and SNA Codes		ii12370	ii11917	ii11611	ii11361	ii11097
SNA Customization	ii12872	ii12388	ii11923	ii11925 ii12008	ii11924 ii12007	ii11092 ii11621 ii12006
SNA Diagnosis	ii12490	ii12389	ii11915	ii11615	ii11357	ii11585
SNA Messages	ii12491	ii12382	ii11916	ii11610	ii11358	ii11096
SNA Network Implementation Guide	ii12487	ii12381	ii11911	ii11609 ii12683	ii11353 ii11493	ii11095
SNA Operation	ii12489	ii12384	ii11914	ii11612	ii11355	ii11098
SNA Migration	ii12486	ii12386	ii11910	ii11614	ii11359	ii11100
SNA Programming		ii12385	ii11920	ii11613	ii11360	ii11099
Quick Reference	ii12500	ii12364	ii11913	ii11616	ii11356	
SNA Resource Definition Reference	ii12488	ii12380 ii12567	ii11912 ii12568	ii11608 ii12569	ii11354 ii12259 ii12570	ii11094 ii11151 ii12260 ii12571
SNA Resource Definition Samples						

Appendix C. Notices

IBM may not offer all of the products, services, or features discussed in this document. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly

tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

This product includes cryptographic software written by Eric Young.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats of unlicensed books and the z/OS Licensed Product Library (LK3T-4307), which contains BookManager and PDF formats of licensed books.

Programming Interfaces

This publication documents information NOT intended to be used as a Programming Interface of z/OS Communications Server.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM	Micro Channel
Advanced Peer-to-Peer Networking	MVS
AFP	MVS/DFP
AD/Cycle	MVS/ESA
AIX	MVS/SP
AIX/ESA	MVS/XA
AnyNet	MQ
APL2	Natural
APPN	NetView
AS/400	Network Station
AT	Nways
BookManager	Notes
BookMaster	NTune
CBPDO	NTuneNCP
C/370	OfficeVision/MVS
CICS	OfficeVision/VM
CICS/ESA	Open Class
C/MVS	OpenEdition
Common User Access	OS/2
C Set ++	OS/390
CT	OS/400
CUA	Parallel Sysplex
DATABASE 2	Personal System/2
DatagLANce	PR/SM
DB2	PROFS
DFSMS	PS/2
DFSMSdfp	RACF
DFSMSHsm	Resource Link
DFSMS/MVS	Resource Measurement Facility
DPI	RETAIN
Domino	RFM
DRDA	RISC System/6000
eNetwork	RMF
Enterprise Systems Architecture/370	RS/6000
ESA/390	S/370
ESCON	S/390
@server	SAA
ES/3090	SecureWay
ES/9000	Slate
ES/9370	SP
EtherStreamer	SP2
Extended Services	SQL/DS
FAA	System/360

FFST	System/370
FFST/2	System/390
FFST/MVS	SystemView
First Failure Support Technology	Tivoli
GDDM	TURBOWAYS
Hardware Configuration Definition	UNIX System Services
IBM	Virtual Machine/Extended Architecture
IBMLink	VM/ESA
IBMLINK	VM/XA
IMS	VSE/ESA
IMS/ESA	VTAM
InfoPrint	WebSphere
Language Environment	XT
LANStreamer	z/Architecture
Library Reader	z/OS
LPDA	zSeries
MCS	400
	3090
	3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

DB2 and NetView are registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the U.S., other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see <http://www.intel.com/tradmarx.htm>.

Other company, product, and service names may be trademarks or service marks of others.

Index

A

- address mapping, IP to SNA 10
- address mask 10
- applying address classification to Sockets-over-SNA routing 23
- assigning IP address to Sockets-over-SNA interface 12
- assigning IP addresses to Sockets-over-SNA nodes 7
- assigning SNA addresses to Sockets-over-SNA 8
- assigning SNA addresses to Sockets-over-SNA nodes
 - assigning LU names 8
 - assigning SNA network names 8
- assigning SNA network IDs to Sockets-over-SNA 8

C

- command syntax
 - ISTSKIFC
 - address operand 22
 - examples of 23
 - netmask operand 23
 - sna0 operand 22
 - ISTSKMAP
 - add operand 18
 - addr_mask argument 20
 - convert operand 19
 - delete operand 19
 - end_ip argument 20
 - flush operand 19
 - get operand 19
 - ip_address argument 19
 - lu_template argument 20
 - qmap operand 19
 - sna_netname argument 20
 - start_ip argument 20
 - syntax of utility 18
- CONNECT_TIMEOUT 28
- conversation, LU 6.2 2
- convert option, and LU name generation 21

D

- destination Sockets-over-SNA 2
- DG_CONV_IDLE_TIMEOUT parameter 28
- DG_SEGMENTATION parameter 28

E

- ENVVAR 27
 - data set 27
 - routing messages 32
 - settings 40
- example network scenarios 3

F

- full-duplex enablement, requirements 6

G

- generating an SNA address 10
- generating LU names, using convert 21

H

- Host Name-to-Address 13
- HOSTNAME parameter 28

I

- IBM SNA Network Registry 8
- installation
 - for MVS 15
- IP address
 - adding line entries to IP-LU mapping table 18
 - assigning to sna0 interface 12, 22
 - classification 23
 - field in IP-LU mapping table 10
 - mapping to SNA 2
- IP-LU mapping table
 - example 11
 - fields 10
 - updating 18
- IP-LU mapping
 - and ISTSKMAP utility 18
 - example 11
 - simplifying 9
 - summary of 12
 - use by Sockets over SNA 11
- ISTSKIFC utility
 - address operand 22
 - examples of using 23
 - netmask operand 23
 - sna0 operand 22
 - syntax for 22
- ISTSKMAP convert option, generating LU names 21
- ISTSKMAP utility
 - add operand 18
 - addr_mask 20
 - convert operand 19
 - convert option 21
 - defining entries when starting Sockets over SNA 11
 - defining LU names 21
 - delete operand 19
 - end_ip argument 20
 - flush operand 19
 - get operand 19
 - ip_address argument 19
 - lu_template 20
 - qmap operand 19
 - sna_netname 20
 - start_ip argument 20
 - syntax 18
 - using with mapping table 18
- ISTSKNST utility
 - display routing tables 40
 - MVS/ESA
 - description of 40
 - output, example 42
 - syntax 41
- ISTSKRTE utility
 - MVS/ESA
 - description of 23

ISTSKRTE utility (*continued*)
 syntax of 24
ISTSKTRC utility
 routing output 44
 syntax of 43

L

LU 6.2 conversation 2
LU name 2
 generating 11, 21
 mapping an address 2
 valid naming conventions 8, 20
LU names, generating using convert option 21
LU template
 field in IP-LU mapping table 10
 limitations 20

M

mapping, IP-LU
 and ISTRKMAP utility 18
 example 11
 simplifying 9
 summary of 12
 use by Sockets over SNA 11
mapping IP addresses to SNA addresses
 generating SNA address 10
 IP-LU mapping process, summary 12
 IP-LU mapping table, example 11
mapping table, IP-LU
 example 11
 fields 10
 updating 18
MAX_DG_CONVS parameter 30
MAX_SENDBUF parameter 30

N

naming convention for LUs 8
netmask operand, ISTRKIFC 23
NetView 6
network address, assigning to sna0 22
network configurations, examples of 3
network ID
 adding entries to IP-LU mapping table 18
 registering (IP) 7
 registering (SNA) 8
Network Information Center (NIC) 7

P

planning for Sockets over SNA
 applying address classification to Sockets-over-SNA
 routing 23
 assigning IP address to Sockets-over-SNA
 interface 12
 assigning IP addresses to Sockets-over-SNA
 nodes 7
 assigning SNA addresses to Sockets-over-SNA
 nodes 8

planning for Sockets over SNA (*continued*)
 mapping IP addresses to Sockets-over-SNA
 addresses 9

R

registering SNA network names 8
requirements
 software 5

S

service information 45
SNA address 8
 generating from an IP address 10
SNA interface, defining for Sockets over SNA 22
SNA network interface value 22
SNA network name
 field in IP-LU mapping table 10
SNA Network Registry 8
sna0 interface, accessing 22
SNMODEn parameter 31
Sockets over SNA 1
 abends, responding to 45
 APPN capability 17
 configuration scenarios 3
 defining
 APPL statements 15
 MODEENT macro entry 16
 SNA networks to 22
 to VTAM 15
 description of 1
 errors, diagnosing 39
 examples 3
 function 1
 increasing the session limit 15
 initialization, order of 35
 managing 6
 mapping IP-LU addresses 9
 MVS/ESA 1
 structure 1
 nodes 7
 assigning IP addresses 7, 12
 assigning SNA addresses 8
 assigning SNA LU names 8
 assigning SNA network names 8
 planning 7
 planning considerations
 and IP addresses 7
 and SNA addresses 7
 mapping IP addresses to SNA addresses 9
 sample network configurations 3
 setting up 7
 software requirements 5
 using IP-LU Mapping Table Entries 11
 using ISTRKTE utility 23
software requirements
 MVS/ESA 5
stopping Sockets over SNA 38
SXMODE_DEFAULT environment variable 31
syntax diagram, reading 47

syntax for utilities

ISTSKIFC

- address operand 22
- examples of 23
- netmask operand 23
- sna0 operand 22

ISTSKMAP

- add operand 18
- addr_mask argument 20
- convert operand 19
- delete operand 19
- end_ip argument 20
- flush operand 19
- get operand 19
- ip_address argument 19
- lu_template argument 20
- qmap operand 19
- sna_netname argument 20
- start_ip argument 20
- syntax of utility 18

T

translation 13

- Host Name-to-Address 13

U

using convert to generate LU names 21

Readers' Comments — We'd Like to Hear from You

z/OS Communications Server
AnyNet Sockets over SNA
Version 1 Release 2

Publication No. SC31-8831-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Cut or Fold
Along Line

|||

[illegible]

Cut or Fold
Along Line



Program Number: 5694-A01



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8831-00



Spine information:



z/OS Communications Server

z/OS V1R2.0 CS: AnyNet Sockets over SNA

Version 1
Release 2